28

Newswire, titled "Nuance Ushers in a New Age of Communications With Voyager Voice

Attached hereto as **Exhibit 17** is a true and correct copy of an article from the PR

27

28

18.

28

KENT DECLARATION EXHIBIT 1



(12) United States Patent Bennett et al.

(10) Patent No.: US 6,633,846 B1 (45) Date of Patent: Oct. 14, 2003

(54) DISTRIBUTED REALTIME SPEECH RECOGNITION SYSTEM

(75) Inventors: Ian M. Bennett, Palo Alto, CA (US); Bandi Ramesh Babu, Anantapur (IN);

Kishor Morkhandikar, Gulbarga (IN); Pallaki Gururaj, Bangalore (IN)

(73) Assignee: **Phoenix Solutions, Inc.**, Palo Alto, CA

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 09/439,145

(22) Filed: Nov. 12, 1999

(51) **Int. Cl.**⁷ **G10L** 15/02; G10L 15/18

(56) References Cited

U.S. PATENT DOCUMENTS

4,473,904 A 9/1984 Suehiro et al.

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	1 094 388 A2	4/2001
EP	1 096 471 A1	5/2001
WO	9811534	3/1998
WO	WO 99/48011 A1	9/1999
WO	WO 99/50830	10/1999
WO	WO 00/14727 A1	3/2000
WO	WO 00/17854 A1	3/2000
WO	WO 00/20962 A2	4/2000
WO	WO 00/21075 A1	4/2000
WO	WO 00/21232 A2	4/2000
WO	WO 00/22610 A1	4/2000
WO	WO 00/30072 A2	5/2000
WO	WO 00/30287 A1	5/2000

WO	WO 00/68823 A2	11/2000
WO	WO 01/16936 A1	3/2001
WO	WO 01/18693 A2	3/2001
WO	WO 01/26093 A1	4/2001
WO	WO 01/78065 A1	10/2001
WO	WO 01/95312 A1	12/2001
WO	WO 02/03380 A1	1/2002

OTHER PUBLICATIONS

G.D. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE*, vol. 73, pp. 268–278, Mar. 1973.

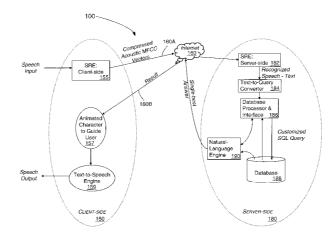
(List continued on next page.)

Primary Examiner—Marsha D. Banks-Harold Assistant Examiner—Martin Lerner (74) Attorney, Agent, or Firm—J. Nicholas Gross

(57) ABSTRACT

A real-time system incorporating speech recognition and linguistic processing for recognizing a spoken query by a user and distributed between client and server, is disclosed. The system accepts user's queries in the form of speech at the client where minimal processing extracts a sufficient number of acoustic speech vectors representing the utterance. These vectors are sent via a communications channel to the server where additional acoustic vectors are derived. Using Hidden Markov Models (HMMs), and appropriate grammars and dictionaries conditioned by the selections made by the user, the speech representing the user's query is fully decoded into text (or some other suitable form) at the server. This text corresponding to the user's query is then simultaneously sent to a natural language engine and a database processor where optimized SQL statements are constructed for a full-text search from a database for a recordset of several stored questions that best matches the user's query. Further processing in the natural language engine narrows the search to a single stored question. The answer corresponding to this single stored question is next retrieved from the file path and sent to the client in compressed form. At the client, the answer to the user's query is articulated to the user using a text-to-speech engine in his or her native natural language. The system requires no training and can operate in several natural languages.

57 Claims, 31 Drawing Sheets



US 6,633,846 B1

Page 2

U.S. PATENT DOCUMENTS

4.587.670 A 5/1986 Levinson et al. 4,783,803 A 11/1988 Baker et al. 4,785,408 A 11/1988 Britton et al. 4,852,170 A 7/1989 Bordeaux 4,914,590 A 4/1990 Loatman et al. 4,991,094 A 2/1991 Fagan et al. 4,991,217 A 2/1991 Garrett et al. 5,068,789 A 11/1991 Van Vliembergen 9/1992 Church 5,146,405 A 5,157,727 A 10/1992 Schloss 5.231.670 A 7/1993 Goldhor et al. 11/1993 Haddock et al. 704/9 5,265,014 A 5,293,584 A 3/1994 Brown et al. 1/1995 Strong 5,384,892 A 5,475,792 A 12/1995 Stanford et al. 5,513,298 A 4/1996 Stanford et al. 5,540,589 A 7/1996 Waters 704/246 5,602,963 A 2/1997 Bissonnette et al. 5,680,628 A 10/1997 Carus et al. 5,727,950 A 3/1998 Cook et al. 4/1998 Flanagan et al. 704/232 5,737,485 A 5,758,322 A 5/1998 Rongley 704/275 5,802,526 A 9/1998 Fawcett et al. 707/104 10/1998 Sarukkai et al. 704/243 5,819,220 A 5,836,771 A 11/1998 Ho et al. 5,867,817 A 2/1999 Catallo et al. 704/255 2/1999 Hansen et al. 5.873.062 A 5,884,302 A 3/1999 Ho 5,915,236 A 6/1999 Gould et al. 704/251 5,934,910 A 8/1999 Ho et al. 5,956,683 A 9/1999 Jacobs et al. 704/275 5,960,394 A 9/1999 Gould et al. 704/240 5,960,399 A 9/1999 Barclay et al. 704/270 5,995,928 A 11/1999 Nguyen et al. 6,009,387 A 12/1999 Ramaswamy et al. 6,029,124 A 2/2000 Gillick et al. 704/200 6.035.275 A 3/2000 Brode et al. 6,112,176 A 8/2000 Goldenthal et al. 6,119,087 A 9/2000 Kuhn et al. 6,138,089 A 10/2000 Guberman 704/207 6,141,640 A 10/2000 Moo 6,144,848 A 11/2000 Walsh et al. 455/419 6,144,938 A 11/2000 Surace et al. 704/257 6,182,038 B1 1/2001 Balakrishnan et al. 6,185,535 B1 2/2001 Hedin et al. 6,233,559 B1 5/2001 Balakrishnan 6,256,607 B1 7/2001 Digalakis et al. 6,269,336 B1 7/2001 Ladd et al. 6,327,561 B1 12/2001 Smith et al. 6,327,568 B1 12/2001 Joost 704/251 6,336,090 B1 1/2002 Chou et al. 704/221 6,363,349 B1 3/2002 Urs et al. 6,374,219 B1 4/2002 Jiang 6,381,594 B1 4/2002 Eichstaedt et al. 6,389,389 B1 5/2002 Meunier et al. 6,408,272 B1 6/2002 White et al. 6,411,926 B1 6/2002 Chang 6,427,063 B1 7/2002 Cook et al. 2001/0016813 A1 8/2001 Brown et al. 2001/0032083 A1 10/2001 Van Cleven 2001/0056346 A1 12/2001 Uevama et al. 2002/0046023 A1 4/2002 Fujii et al. 2002/0059068 A1 5/2002 Rose et al. 2002/0059069 A1 5/2002 Hsu et al. 2002/0086269 A1 7/2002 Shpiro 2002/0087325 A1 7/2002 Lee et al. 2002/0087655 A1 7/2002 Bridgman et al.

2002/0091527 A1

7/2002 Shiau

OTHER PUBLICATIONS

- J.H. Baker, "The Dragon System-An Overview," IEEE Trans. on ASSP Processing, ASSP-23(1): 24-29, Feb. 1975. I. Bennett, "A Study of Speech Compression Using Analog Time Domain Sampling techniques," A Dissertation Sub-
- mitted to the Dept. Of Electrical Engineering and the Committee on Graduate Studies of Stanford University, May 1975, pp. 16-32; 76-111.
- H.R. Rabiner, "Digital Processing of Speech Signals," Prentice Hall, 1978, pp. 116-171; 355-395.
- F. Jelinek et al, "Continuous Speech Recognition: Statistical Methods," Handbook of Statistics, vol. 2, P.R. Krishnaiah, Ed. Amsterdam, The Netherlands, North-Holland, 1982, pp. 549-573.
- L.R. Bahl, F. Jeninek, R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-5, pp. 179-190, Mar. 1983.
- R.A. Hudson, "Word Grammar," Blackman Inc., Cambridge, MA, 1984, pp. 1-14; 41-42; 76-90; 94-98; 106-109; 211-220.
- R. Quirk, S. Greenbaum, G. Leech and J. Svartvik, "A Comprehensive Grammar of English Language," Longman, London and New York, 1985, pp. 245-331.
- J. Makhoul, S. Roucos, H. Gish, "Vector Quantization in Speech Coding," Proceedings of the IEEE, vol. 73, No. 11, Nov. 1985, pp. 1551-1588.
- L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of the IEEE, vol. 77, No. 2, Feb. 1989, pp. 257-286.
- A. Gersho and R.M. Gray, "Vector Quantization and Compression," Kluwer Academic Publishers, 1991, pp. 309–340.
- H.R. Rabiner and B.H. Juang, "Fundamentals of Speech Recognition," Prentice Hall, 1993, pp. 11-68.
- Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco, "Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition," Journal of Pattern Recognition and Artificial Intelligence, vol. 7, No. 4, 1993, pp. 899-916.
- Arons, B., "The Design of Audio Servers and Toolkits for Supporting Speech in the User Interface," believed to be published in: Journal of the American Voice I/O Society, pp. 27-41, Mar. 1991.
- Hazen, T et al., "Recent Improvements in an Approach to Segment-Based Automatic Language Identification," believed to be published in: Proceedings of the 1994 International Conference on Spoken Language Processing, Yokohama, Japan, pp. 1883-1886, Sep. 1994.
- House, D., "Spoken-Language Access to Multimedia (SLAM): A Multimodal Interface to the World-Wide Web,' Masters Thesis, Oregon Graduate Institute, Department of Computer Science & Engineering, 59 pages, Apr. 1995.
- Julia, L. et al., "HTTP://WWW.SPEECH.SRI.COM/ DEMOS/ ATIS.HTML," believed to be published in: Proceedings AAAI'97: Stanford, pp. 72-76, Jul. 1997.
- Lau, R. et al, "Webgalaxy-Integrating Spoken Language and Hypertext Navigation," believed to be published in: in Kokkinakis, G. et al., (Eds.) Eurospeech '97, Proceedings of the 5th European Conference on Speech Communication and Technology, Rhodes (Greece), Sep. 22-25, 1997. pp. 883-886, 1997.

US 6,633,846 B1

Page 3

Digalakis, V. et al., "Product-Code Vector Quantization of Cepstral Parameters for Speech Recognition over the WWW," believed to be published in: Proc. ICSLP '98, 4 pages, 1998.

Melin, H., "On Word Boundary Detection in Digit-Based Speaker Verification," believed to be published in: Workshop on Speaker Recognition and its Commercial and Forensic Applications (RLA2C), Avignon, France, Apr. 20–23, pp. 46–49, 1998.

Ramaswamy, G. et al., "Compression of Acoustic Features for Speech Recognition in Network Environments," believed to be published in: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 977–980, Jun. 1998.

Lu, B. et al., "Scalability Issues in the Real Time Protocol (RTP)," Project Report for CPSC 663 (Real Time Systems), Dept. of Computer Science, Texas A & M University, 19 pages, 1999.

Giuliani, D. et al., "Training of HMM with Filtered Speech Material for Hands-Free Recognition," believed to be published in: Proceedings of ICASSP '99, Phoenix, USA, 4 pages, 1999.

Digalakis, V. et al., "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web," believed to be published in: IEEE Journal on Selected Areas of Communications, 22 pages, 1999.

Tsakalidis, S. et al., "Efficient Speech Recognition Using Subvector Quantization and Discrete-Mixture HMMs," believed to be published in: Proc. ICASSP '99, 4 pages, 1999

Lin, B. et al., "A Distributed Architecture for Cooperative Spoken Dialogue Agents with Coherent Dialogue State and History," believed to be published in: IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado, USA, 4 pages, Dec. 1999.

Meunier, J., "RTP Payload Format for Distrubuted Speech Recognition," 48th IETF AVT WG—Aug. 3, 2000, 10 pages, 2000.

Sand Cherry Networks, SoftServer product literature, 2 pages, 2001.

Kim,H. et al., "A Bitstream-Based Front-End for Wireless Speech Recognition on IS-136 Communications System," IEEE Transactions on Speech and Audio Processing, vol. 9, No. 5, pp. 558-568, Jul. 2001.

- L. Travis, "Handbook of Speech Pathology," Appleton-Century-Crofts, Inc., 1957, pp. 91–124.
- L.E. Baum, T. Petrie, "Statistical Inference for Probabilistic Functions for Finite State Markov Chains," *The Annals of Mathematical Statistics*, 37: 1554–1563, 1966.
- P. Lieberman, "Intonation, Perception and Language," Research Monograph No. 38, MIT Press, Cambridge, Mass., 1967, pp. 5–37.
- L.E. Baum et al., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics*, 1970, vol. 41, No. 1, pp. 164–171.
- J.L. Flanagan, "Speech Analysis Synthesis and Perception," 2nd edition, Springer–Verlag Berlin, 1972, pp. 1–53.
- L.E. Baum, "An Inequality and Associated Maximation Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," *Inequalities*–III, pp. 1–8, 1972.

^{*} cited by examiner

U.S. Patent Oct. 14, 2003 Sheet 1 of 31 US 6,633,846 B1

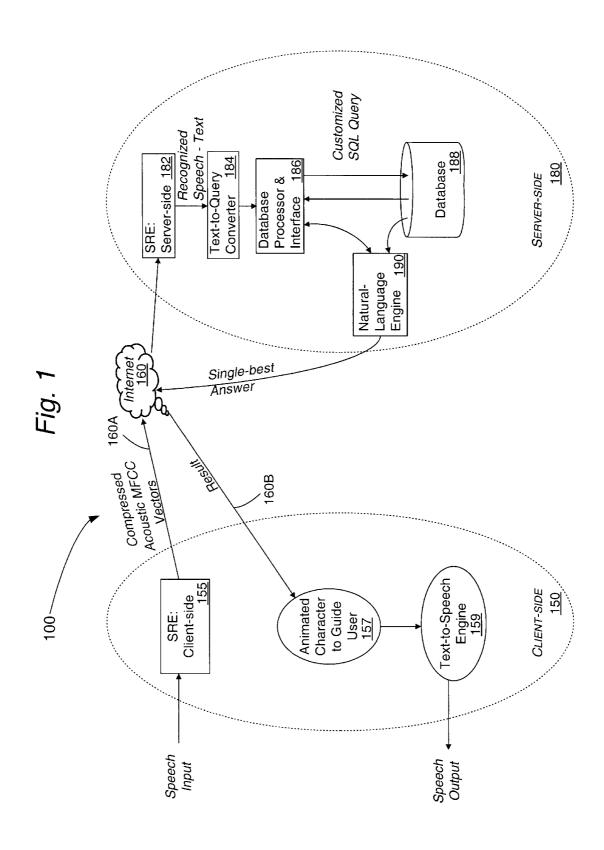


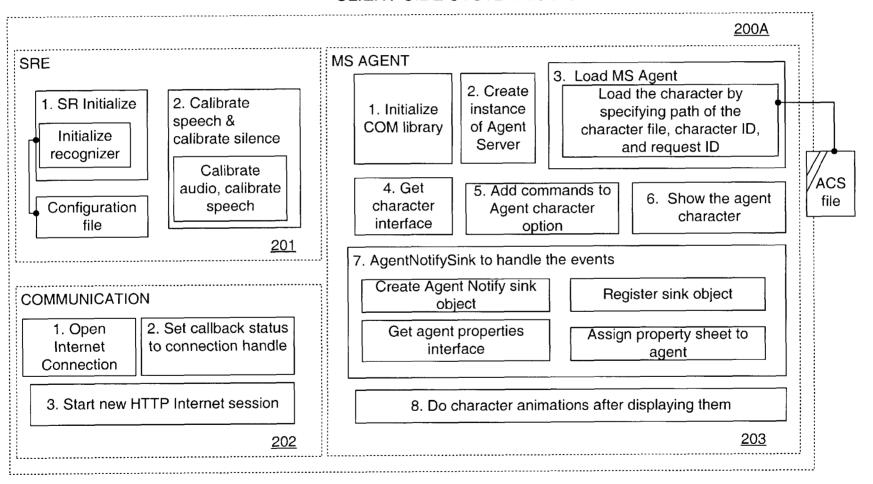
Figure 2A

CLIENT-SIDE SYSTEM LOGIC

U.S. Patent

Oct. 14, 2003

Sheet 2 of 31



U.S. Patent Oct. 14, 2003 Sheet 3 of 31 US 6,633,846 B1

Figure 2B CLIENT-SIDE SYSTEM LOGIC

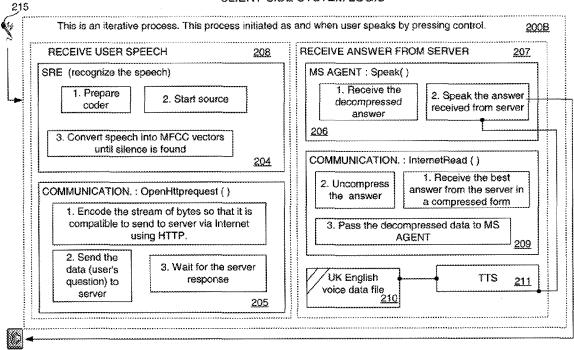
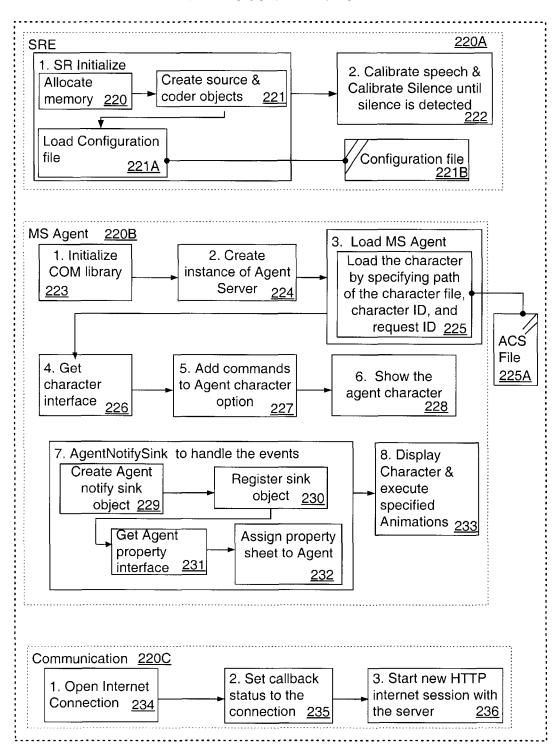


Figure 2C CLIENT-SIDE SYSTEM LOGIC

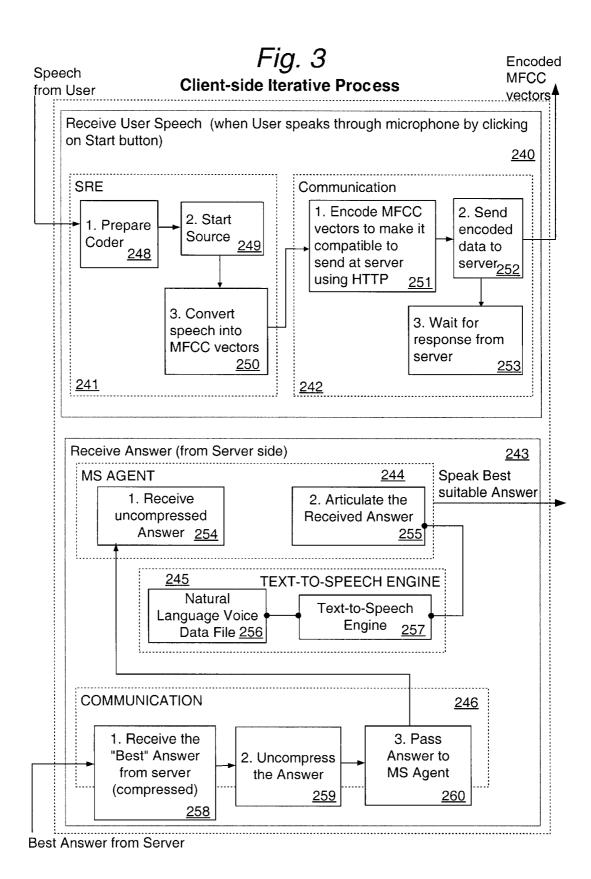
UN-INITIALIZATION (performs as and when user guits i.e. closes the web page) 200C COMMUNICATION MS AGENT SRE 213 212 2. Release 1. Release 3. Unload 1. Close the Internet commands character agent handle i.e. the Interface Interface a. Delete the objects connection created while established with initialization process server. 4. Release 5. Release b. Deallocate the 6. Unregister AgentNotifysink prop. sheet Agent Notifysink memory assigned to Interface Interface the structure, which will be holding the 2. Close the Internet parameters for session which is speech created at the time of initialization 7. Release Agent Interface <u>214</u>

U.S. Patent Oct. 14, 2003 Sheet 5 of 31 US 6,633,846 B1

Fig. 2D Client-side Initialization



U.S. Patent Oct. 14, 2003 Sheet 6 of 31 US 6,633,846 B1



U.S. Patent Oct. 14, 2003 Sheet 7 of 31

Fig. 4
Client-side Un-Initialization

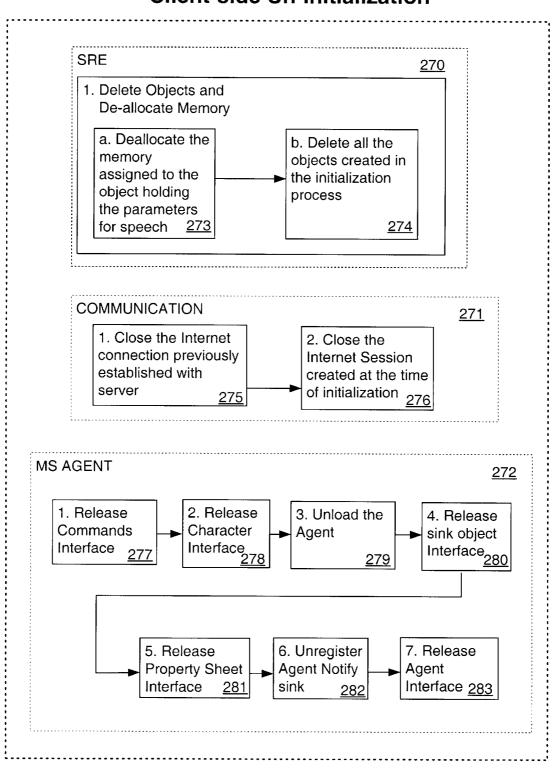


Fig. 4A

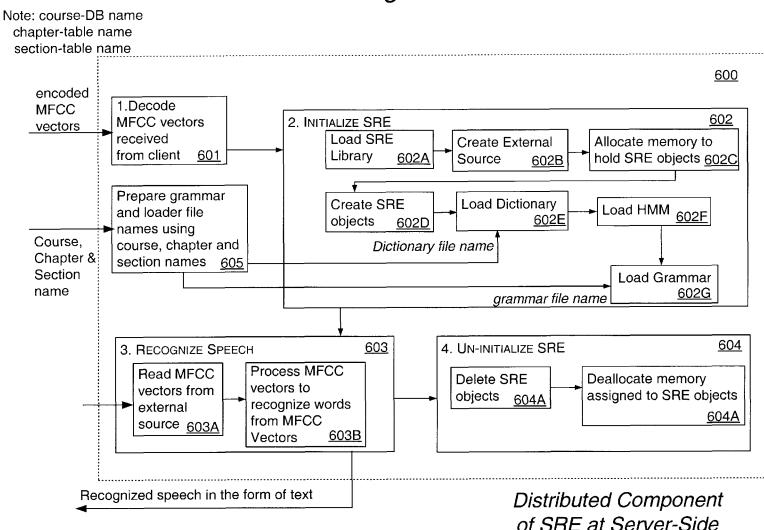
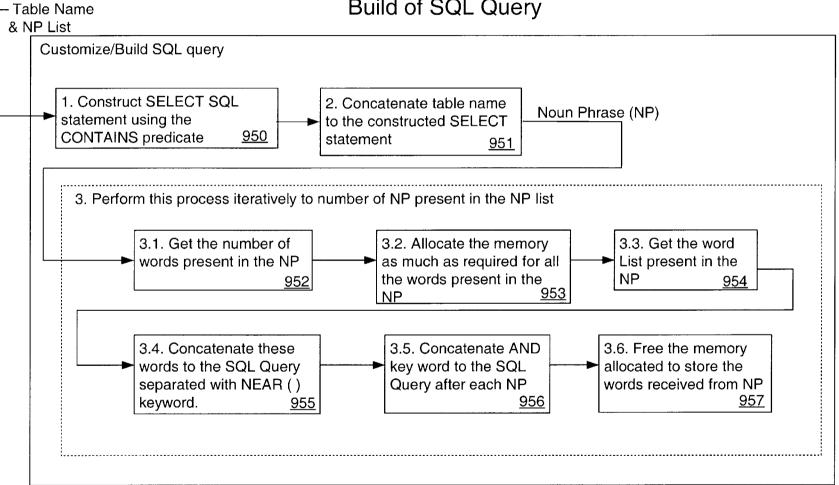


Fig. 4B **Build of SQL Query** Noun Phrase (NP) 951

U.S. Patent

Oct. 14, 2003

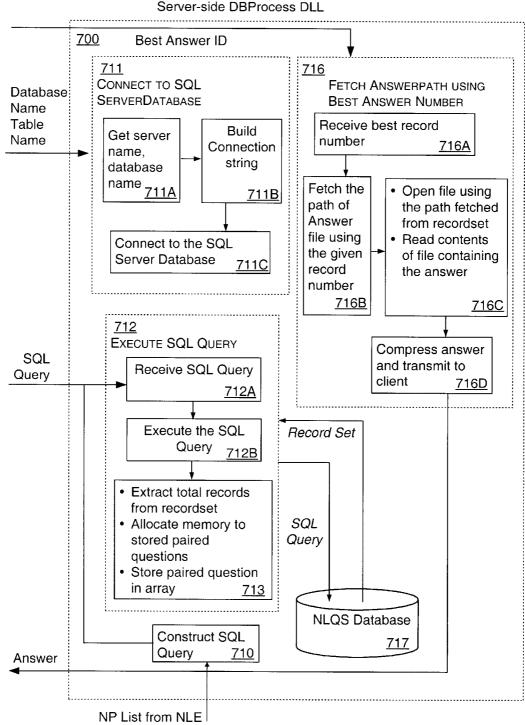
Sheet 9 of 31



U.S. Patent Oct. 14, 2003

Sheet 10 of 31

Fig. 4C



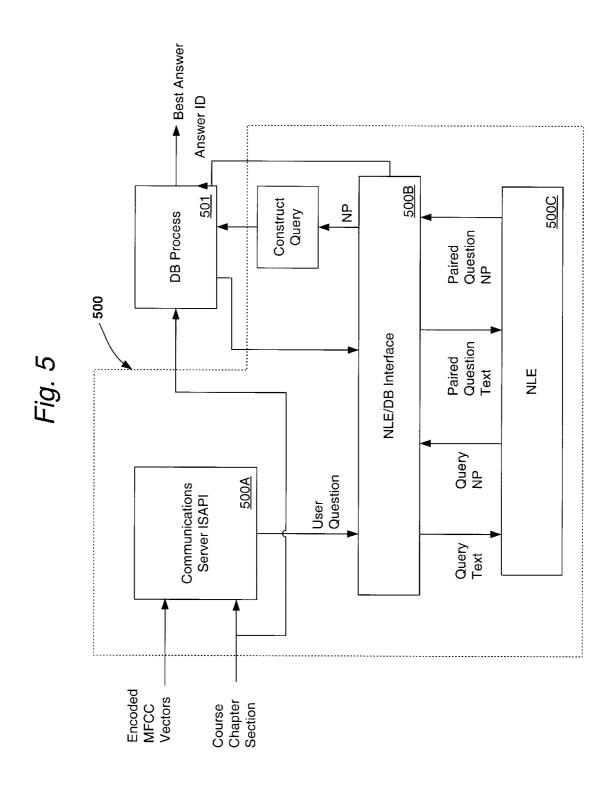
U.S. Patent Oct. 14, 2003 Sheet 11 of 31 US 6,633,846 B1

Fia. 4D Note: PQ - Paired Question Interface Logic between NP- Noun Phrase NLE and DBProcess.DLL Best Red Line - I / O Answer NP list of Number Paired Questions from DB PQ813 GET NP LIST GET BEST ANSWER ID FOR PAIRED 880 GET NP LIST FOR QUESTION **≜**Best THE USER'S QUESTION Answer NP List from Receive the Number Receive the Question Get the PQs from question NP list and PQ DBProcess.dll from client using 815A Compare NP list 813A **NLE** 880A 880B Compare NP of user's question with Get NP List PQ from DB to find List using NLE Question out the best suitable <u>813B</u> Ą question present in NP List of the DB NP List User's Question Question Paired Questions 9b. Tokenize 9c. Tag all the 900 INITIALIZE GROUPER the words from **RESOURCES** tokens 909C the given text Initialize Initialize <u>909</u>B Token Tagger Resources Resources 900A 900B 9d. Group all tagged tokens Initialize Create to form the NP Grouper Grouper 909D resources 900C <u>900D</u> 9E. UN-INITIALIZE GROUPER RESOURCES OBJECT AND FREE THE RESOURCES Free token Free grouper Free tagger **NLE** resources resources resources 909EC 909EA <u>909EB</u>

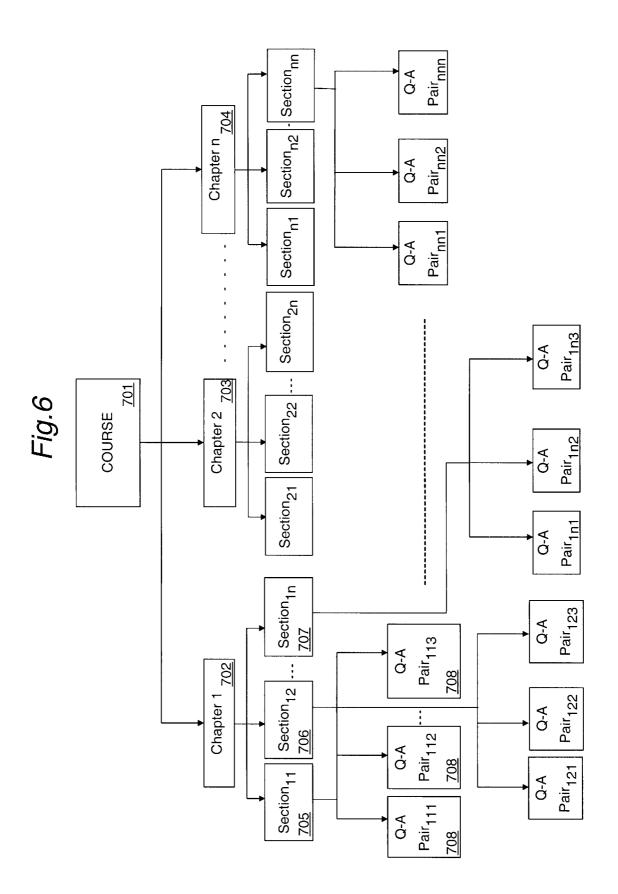
U.S. Patent

Oct. 14, 2003

Sheet 12 of 31



U.S. Patent Oct. 14, 2003 Sheet 13 of 31 US 6,633,846 B1



U.S. Patent

Oct. 14, 2003

Sheet 14 of 31

US 6,633,846 B1

FIELD NAME	DATA TYPE	SIZE	NULL	PRIMARY KEY	INDEXED?
701A	702A	703A	704A	705A	706A
ChapterName 707A	Varchar	255	No	No	Yes
SectionName	Varchar	255	No	No	Yes
708A					

Fig.7B

FIELD NAME 720	D ата Түре <u>721</u>	Size <u>722</u>	Null <u>723</u>	PRIMARY KEY <u>724</u>	INDEXED? <u>725</u>
Chapter_ID <u>726</u>	Integer		No	Yes	Yes
Answer_ID <u>727</u>	Char	5	No	UNIQUE	Yes
Section_Name <u>728</u>	Varchar	255	No	UNIQUE	Yes
Answer_Title 729	Varchar	255	Yes	No	Yes
PairedQuestion 730	Text	16	No	No	Yes (Full-Text)
AnswerPath 731	Varchar	255	No	No	Yes
Creator <u>732</u>	Varchar	50	No	No	Yes
Date_of_Creation <u>733</u>	Date	-	No	No	Yes
Date_of_Modification 734	Date	-	No	No	Yes

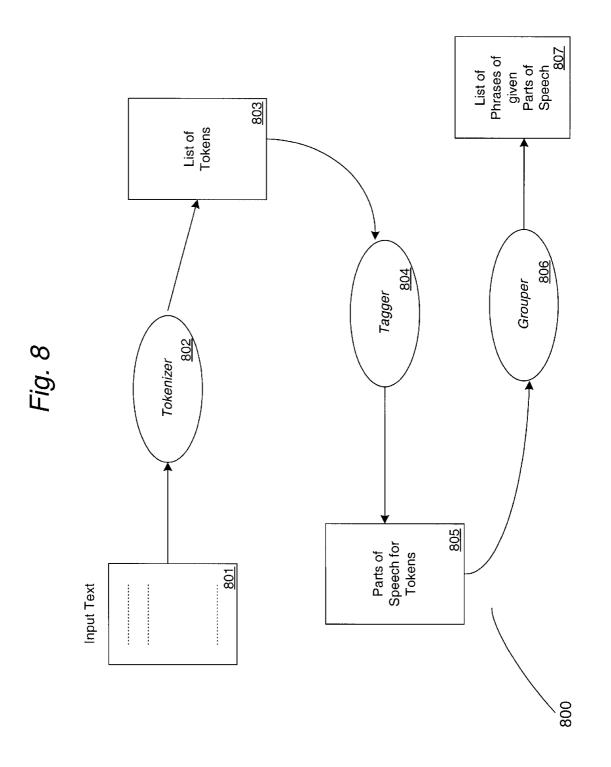
Fig. 7C

Field	<u>720</u>	Description 735
AnswerID	<u>727</u>	An integer - automatically incremented for user convenience
Section_Name	<u>728</u>	Name of section to which the particular record belongs. This field along with AnswerlD has to be made primary key
Answer_Title	<u>729</u>	A short description of the answer
PairedQuestion	<u>730</u>	Contains one or more combinations of questions for the related answer whose path is stored in the next column AnswerPath
AnswerPath	<u>731</u>	Contains the path of text file, which contains the answer to the related questions stored in the previous column
Creator	<u>732</u>	Name of content creator
Date_of_Creation	<u>733</u>	Date on which content has been added
Date_of_Modification	<u>734</u>	Date on which content has been changed or modified

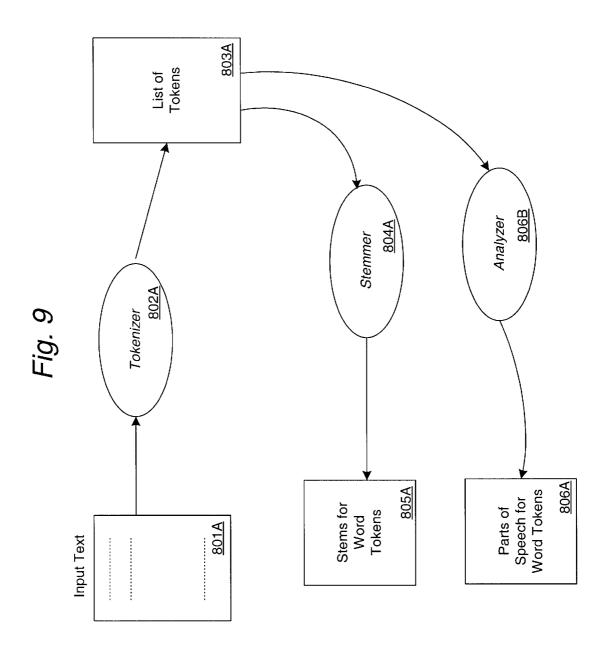
Fig. 7D

FIELD <u>740</u>	Dата Түре <u>741</u>	Size <u>742</u>	Null <u>743</u>	Primary Key <u>744</u>	INDEXED <u>745</u>
Answer_ID 746	Char	5	No	Yes	Yes
Answer_Title 747	Varchar	255	Yes	No	No
PairedQuestion 748	Text	16	No	No	Yes (Full-Text)
Answer_Path 749	Varchar	255	No	No	No
Creator 750	Varchar	50	No	No	No
Date_of_Creation 751	Date	-	No	No	No
Date_of_Modification 752	Date	-	No	No	No

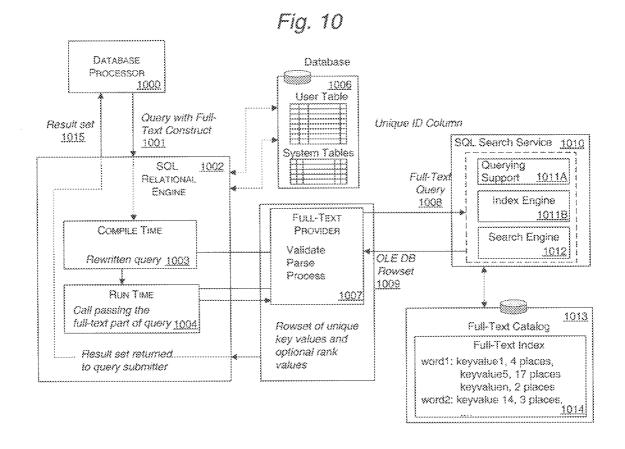
U.S. Patent Oct. 14, 2003 Sheet 18 of 31 US 6,633,846 B1



U.S. Patent Oct. 14, 2003 Sheet 19 of 31 US 6,633,846 B1



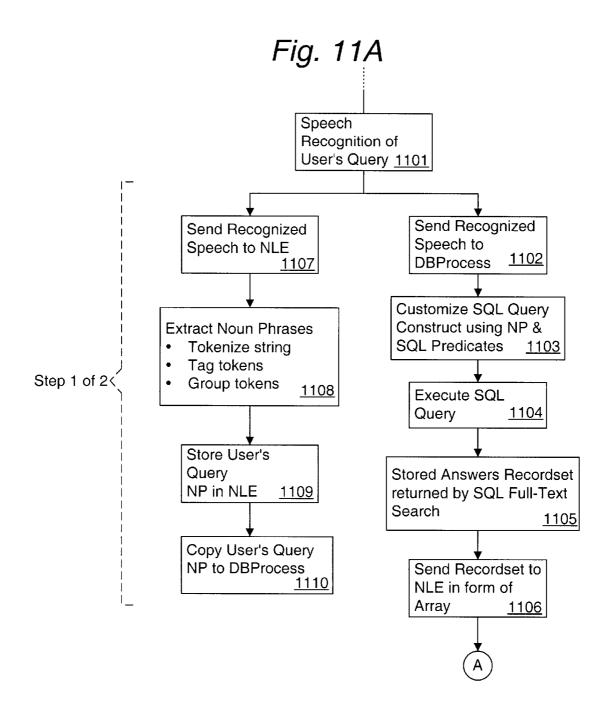
U.S. Patent Oct. 14, 2003 Sheet 20 of 31 US 6,633,846 B1



U.S. Patent

Oct. 14, 2003

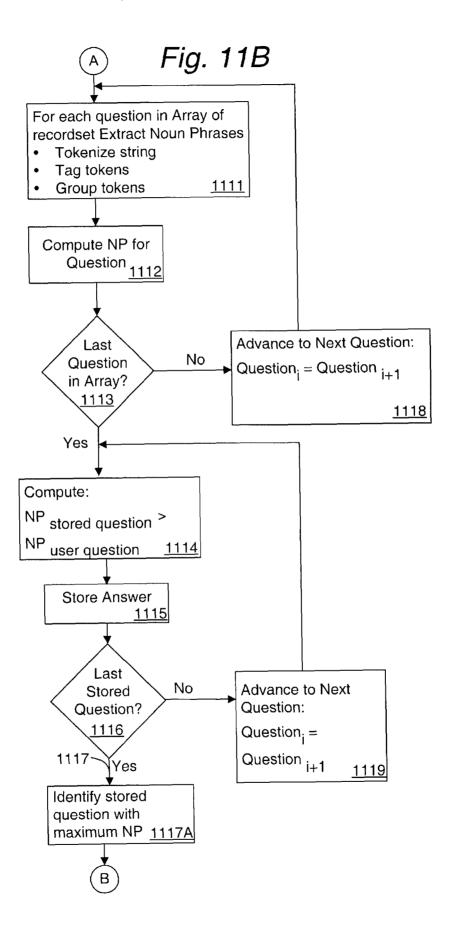
Sheet 21 of 31



U.S. Patent

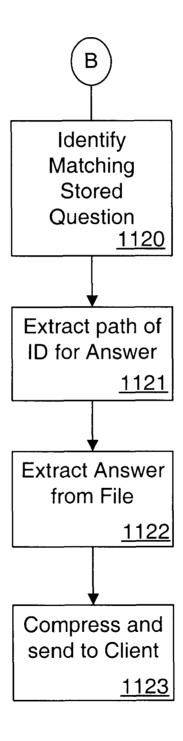
Oct. 14, 2003

Sheet 22 of 31

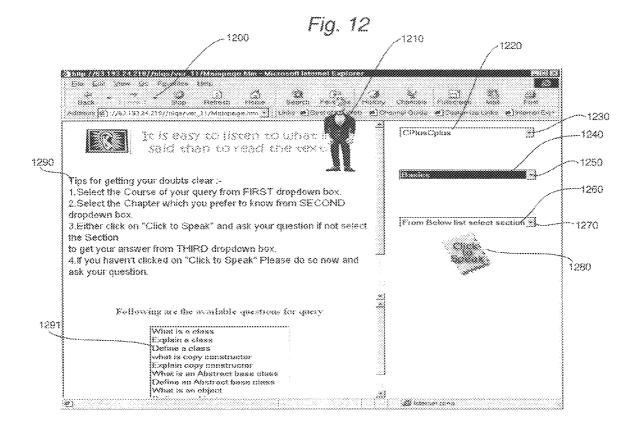


U.S. Patent Oct. 14, 2003 Sheet 23 of 31 US 6,633,846 B1

Fig. 11C

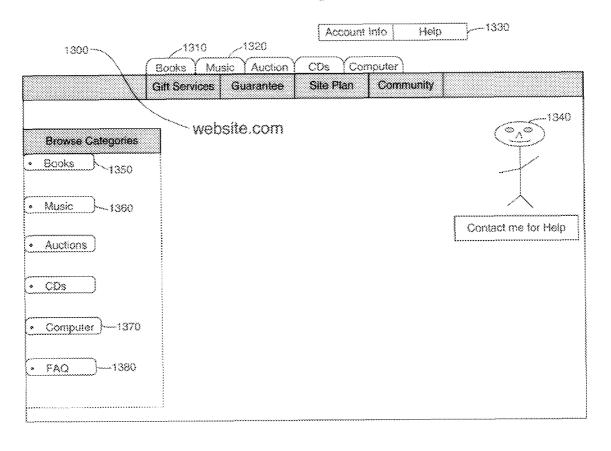


U.S. Patent Oct. 14, 2003 Sheet 24 of 31 US 6,633,846 B1



U.S. Patent Oct. 14, 2003 Sheet 25 of 31 US 6,633,846 B1

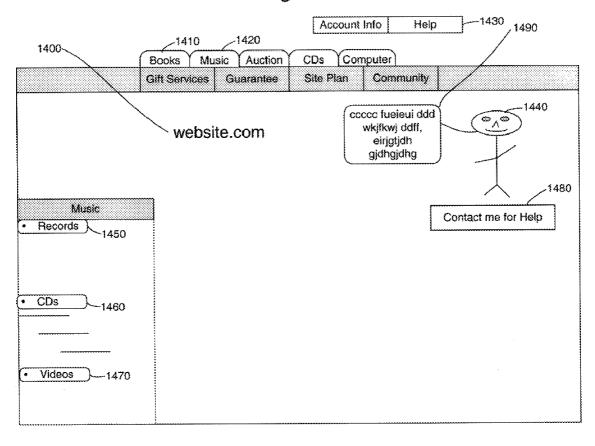
Fig. 13



U.S. Patent Oct. 14, 2003

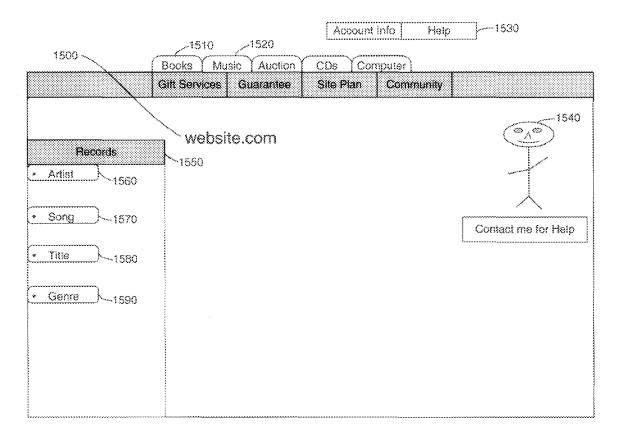
Sheet 26 of 31

Fig. 14



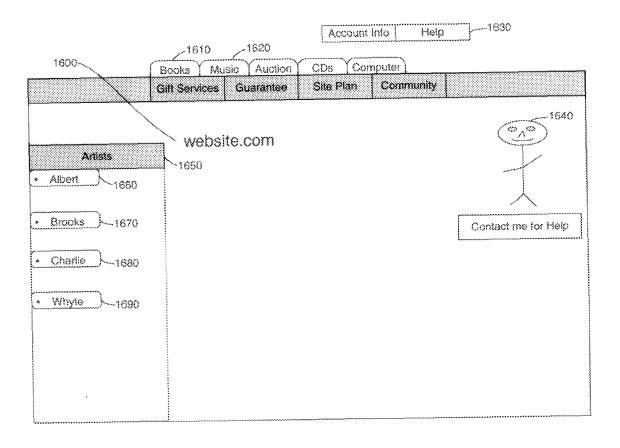
U.S. Patent Oct. 14, 2003 Sheet 27 of 31 US 6,633,846 B1

Fig. 15



U.S. Patent Oct. 14, 2003 Sheet 28 of 31 US 6,633,846 B1

Fig. 16



U.S. Patent Oct. 14, 2003 Sheet 29 of 31 US 6,633,846 B1

Fig. 17

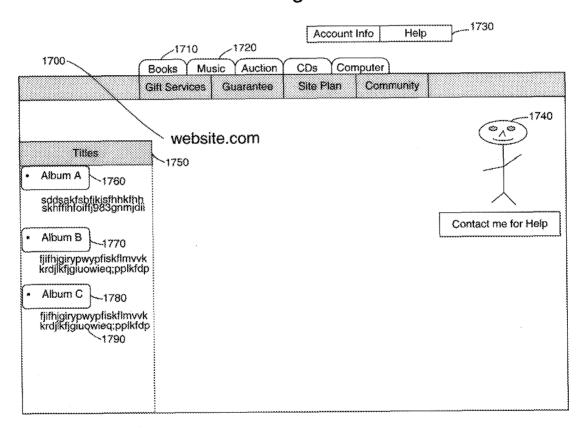
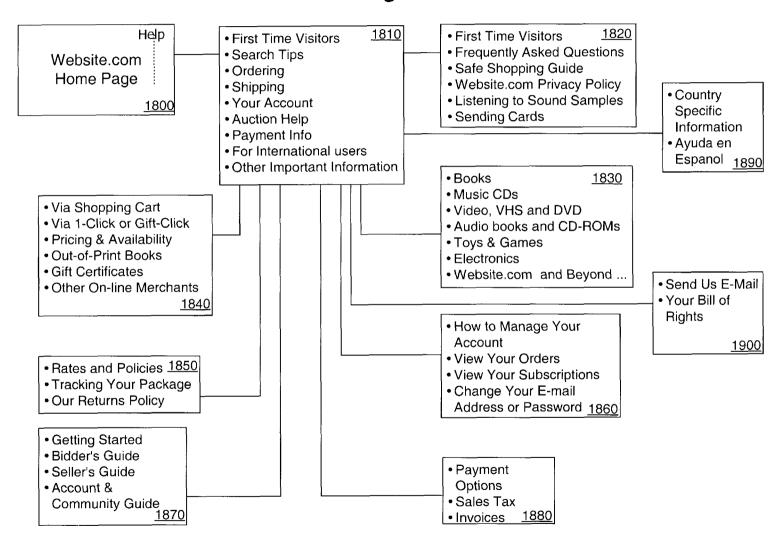


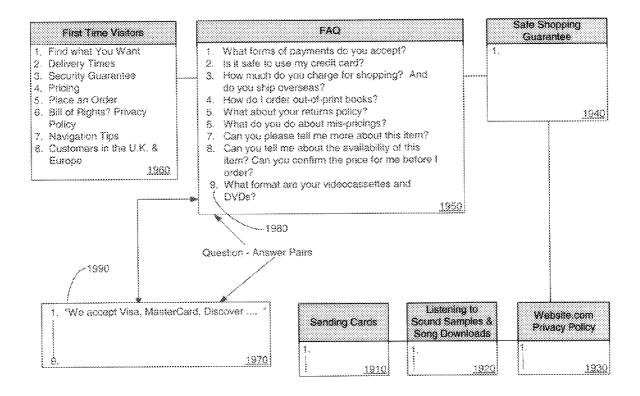
Fig. 18A

Document 52-2



U.S. Patent Oct. 14, 2003 Sheet 31 of 31 US 6,633,846 B1

Fig. 18B



1

DISTRIBUTED REALTIME SPEECH RECOGNITION SYSTEM

RELATED APPLICATIONS

The present application is related to the following applications also filed contemporaneously herewith:

- 1) Ser. No. 09/439,173 entitled Speech Based Learning/ Training system, attorney docket no. PHO 99-002;
- Ser. No. 09/439,174 entitled Internet Server with 10 Speech Support for Enhanced Interactivity—attorney docket no. PHO 99-003;
- Ser. No. 09/439,060 entitled Intelligent Query Engine For Processing Voice Based Queries—attorney docket no. PHO 99-004;

The above are incorporated by reference herein.

FIELD OF THE INVENTION

The invention relates to a system and an interactive method for responding to speech based user inputs and queries presented over a distributed network such as the INTERNET or local intranet. This interactive system when implemented over the World-Wide Web services (WWW) of the INTERNET, functions so that a client or user can ask a question in a natural language such as English, French, German, Spanish or Japanese and receive the appropriate answer at his or her computer or accessory also in his or her native natural language. The system has particular applicability to such applications as remote learning, e-commerce, technical e-support services, INTERNET searching, etc.

BACKGROUND OF THE INVENTION

The INTERNET, and in particular, the World-Wide Web (WWW, is growing in popularity and usage for both commercial and recreational purposes, and this trend is expected to continue. This phenomenon is being driven, in part, by the increasing and widespread use of personal computer systems and the availability of low cost INTERNET access. The emergence of inexpensive INTERNET access devices and high speed access techniques such as ADSL, cable modems, satellite modems, and the like, are expected to further accelerate the mass usage of the WWW.

Accordingly, it is expected that the number of entities offering services, products, etc., over the WWW will 45 increase dramatically over the coming years. Until now, however, the INTERNET "experience" for users has been limited mostly to non-voice based input/output devices, such as keyboards, intelligent electronic pads, mice, trackballs, printers, monitors, etc. This presents somewhat of a bottle- 50 neck for interacting over the WWW for a variety of reasons.

First, there is the issue of familiarity. Many kinds of applications lend themselves much more naturally and fluently to a voice-based environment. For instance, most people shopping for audio recordings are very comfortable 55 with asking a live sales clerk in a record store for information on titles by a particular author, where they can be found in the store, etc. While it is often possible to browse and search on one's own to locate items of interest, it is usually easier and more efficient to get some form of human assistance first, and, with few exceptions, this request for assistance is presented in the form of a oral query. In addition, many persons cannot or will not, because of physical or psychological barriers, use any of the aforementioned conventional I/O devices. For example, many older 65 persons cannot easily read the text presented on WWW pages, or understand the layout/hierarchy of menus, or

2

manipulate a mouse to make finely coordinated movements to indicate their selections. Many others are intimidated by the look and complexity of computer systems, WWW pages, etc., and therefore do not attempt to use online services for this reason as well.

Thus, applications which can mimic normal human interactions are likely to be preferred by potential on-line shoppers and persons looking for information over the WWW. It is also expected that the use of voice-based systems will increase the universe of persons willing to engage in e-commerce, e-learning, etc. To date, however, there are very few systems, if any, which permit this type of interaction, and, if they do, it is very limited. For example, various commercial programs sold by IBM (VIAVOICETM) and Kurzweil (DRAGONTM) permit some user control of the interface (opening, closing files) and searching (by using previously trained URLs) but they do not present a flexible solution that can be used by a number of users across multiple cultures and without time consuming voice training. Typical prior efforts to implement voice based functionality in an INTERNET context can be seen in U.S. Pat. No. 5,819,220 incorporated by reference herein.

Another issue presented by the lack of voice-based systems is efficiency. Many companies are now offering technical support over the INTERNET, and some even offer live operator assistance for such queries. While this is very advantageous (for the reasons mentioned above) it is also extremely costly and inefficient, because a real person must be employed to handle such queries. This presents a practical limit that results in long wait times for responses or high labor overheads. An example of this approach can be seen U.S. Pat. No. 5,802,526 also incorporated by reference herein. In general, a service presented over the WWW is far more desirable if it is "scalable," or, in other words, able to handle an increasing amount of user traffic with little if any perceived delay or troubles by a prospective user.

In a similar context, while remote learning has become an increasingly popular option for many students, it is practically impossible for an instructor to be able to field questions from more than one person at a time. Even then, such interaction usually takes place for only a limited period of time because of other instructor time constraints. To date, however, there is no practical way for students to continue a human-like question and answer type dialog after the learning session is over, or without the presence of the instructor to personally address such queries.

Conversely, another aspect of emulating a human-like dialog involves the use of oral feedback. In other words, many persons prefer to receive answers and information in audible form. While a form of this functionality is used by some websites to communicate information to visitors, it is not performed in a real-time, interactive question-answer dialog fashion so its effectiveness and usefulness is limited.

Yet another area that could benefit from speech-based interaction involves so-called "search" engines used by INTERNET users to locate information of interest at web sites, such as the those available at YAHOO®.com, METACRAWLER®.com, EXCITE®.com, etc. These tools permit the user to form a search query using either combinations of keywords or metacategories to search through a web page database containing text indices associated with one or more distinct web pages. After processing the user's request, therefore, the search engine returns a number of hits which correspond, generally, to URL pointers and text excerpts from the web pages that represent the closest match made by such search engine for the particular user query

3

based on the search processing logic used by search engine. The structure and operation of such prior art search engines, including the mechanism by which they build the web page database, and parse the search query, are well known in the art. To date, applicant is unaware of any such search engine 5 that can easily and reliably search and retrieve information based on speech input from a user.

There are a number of reasons why the above environments (e-commerce, e-support, remote learning, INTER-NET searching, etc.) do not utilize speech-based interfaces, 10 despite the many benefits that would otherwise flow from such capability. First, there is obviously a requirement that the output of the speech recognizer be as accurate as possible. One of the more reliable approaches to speech recognition used at this time is based on the Hidden Markov Model (HMM)—a model used to mathematically describe any time series. A conventional usage of this technique is disclosed, for example, in U.S. Pat. No. 4,587,670 incorporated by reference herein. Because speech is considered to have an underlying sequence of one or more symbols, the 20 HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of 25 probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O, is generated with probability density B_i(O_i). It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model. The basic theory of HMMs was published in a series of classic papers by Baum and his 35 colleagues in the late 1960's and early 1970's. HMMs were first used in speech applications by Baker at Carnegie Mellon, by Jelenik and colleagues at IBM in the late 1970's and by Steve Young and colleagues at Cambridge University, UK in the 1990's. Some typical papers and texts are as follows:

- L. E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains", Ann. Math. Stat., 37: 1554–1563, 1966
- L. E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1–8, 1972
- 3. J. H. Baker, "The dragon system—An Overview", IEEE Trans. on ASSP Proc., ASSP-23(1): 24–29, Feb. 1975
- F. Jeninek et al, "Continuous Speech Recognition: Statistical methods" in Handbook of Statistics, II, P. R. Kristnaiad, Ed. Amsterdam, The Netherlands, North-Holland. 1982
- L. R. Bahl, F. Jeninek, R. L. Mercer, "A maximum likelihood approach to continuous speech recognition", IEEE Trans. Pattern Anal. Mach. Intell., PAMI-5: 179–190,1983
- J. D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, N.J. 1980.
- H. R. Rabiner and B. H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, 1993
- 8. H. R. Rabiner, "Digital Processing of Speech Signals", Prentice Hall, 1978

4

More recently research has progressed in extending HMM and combining HMMs with neural networks to speech recognition applications at various laboratories. The following is a representative paper:

 Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco (1993), Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition. Journal of Pattern Recognition and Artificial Intelligence, Vol.7, No.4 pp. 899–916. Also in I. Guyon and P. Wang editors, Advances in Pattern Recognition Systems using Neural Networks, Vol. 7 of a Series in Machine Perception and Artificial Intelligence. World Scientific, Feb. 1994.

All of the above are hereby incorporated by reference. While the HMM-based speech recognition yields very good results, contemporary variations of this technique cannot guarantee a word accuracy requirement of 100% exactly and consistently, as will be required for WWW applications for all possible all user and environment conditions. Thus, although speech recognition technology has been available for several years, and has improved significantly, the technical requirements have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily.

In contrast to word recognition, Natural language processing (NLP) is concerned with the parsing, understanding and indexing of transcribed utterances and larger linguistic units. Because spontaneous speech contains many surface phenomena such as disfluencies,—hesitations, repairs and restarts, discourse markers such as 'well' and other elements which cannot be handled by the typical speech recognizer, it is the problem and the source of the large gap that separates speech recognition and natural language processing technologies. Except for silence between utterances, another problem is the absence of any marked punctuation available for segmenting the speech input into meaningful units such as utterances. For optimal NLP performance, these types of phenomena should be annotated at its input. However, most continuous speech recognition systems produce only a raw sequence of words. Examples of conventional systems using NLP are shown in U.S. Pat. Nos. 4,991,094, 5,068,789, 5,146,405 and 5,680,628, all of which are incorporated by

Second, most of the very reliable voice recognition systems are speaker-dependent, requiring that the interface be "trained" with the user's voice, which takes a lot of time, and is thus very undesirable from the perspective of a WWW environment, where a user may interact only a few times with a particular website. Furthermore, speaker-dependent systems usually require a large user dictionary (one for each unique user) which reduces the speed of recognition. This makes it much harder to implement a real-time dialog interface with satisfactory response capability (i.e., something that mirrors normal conversation—on the order of 3-5 seconds is probably ideal). At present, the typical shrinkwrapped speech recognition application software include offerings from IBM (VIAVOICETM) and Dragon Systems (DRAGONTM). While most of these applications are adequate for dictation and other transcribing applications, they are woefully inadequate for applications such as NLQS where the word error rate must be close to 0%. In addition these offerings require long training times and are typically are non client-server configurations. Other types of trained systems are discussed in U.S. Pat. No. 5,231,670 assigned to Kurzweil, and which is also incorporated by reference herein.

Case 3:08-cv-00863-MHP

Another significant problem faced in a distributed voicebased system is a lack of uniformity/control in the speech recognition process. In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. A well-known system of this type is depicted in U.S. Pat. No. 4,991,217 incorporated by reference herein. These clients can take numerous forms (desktop PC, laptop PC, PDA, etc.) having varying speech signal processing and communications capability. Thus, from the server side perspective, it is not easy to assure uniform treatment of all users accessing a voice-enabled web page, since such users may have significantly disparate word recognition and error rate performances. While a prior art reference to Gould et al.—U.S. Pat. No. 5,915,236—discusses generally the notion of tailoring a recognition process to a set of available computational resources, it does not address or attempt to solve the issue of how to optimize resources in a distributed environment such as a client-server model. Again, to enable such voice-based technologies on a wide-spread scale it is far more preferable to have a system that harmonizes and accounts for discrepancies in individual systems so that even 20 the thinnest client is supportable, and so that all users are able to interact in a satisfactory manner with the remote server running the e-commerce, e-support and/or remote learning application.

Two references that refer to a distributed approach for speech recognition include U.S. Pat. Nos. 5,956,683 and 5,960,399 incorporated by reference herein. In the first of these, U.S. Pat. No. 5,956,683—Distributed Voice Recognition System (assigned to Qualcomm) an implementation of a distributed voice recognition system between a telephony-based handset and a remote station is described. In this implementation, all of the word recognition operations seem to take place at the handset. This is done since the patent describes the benefits that result from locating of the system for acoustic feature extraction at the portable or cellular phone in order to limit degradation of the acoustic 35 features due to quantization distortion resulting from the narrow bandwidth telephony channel. This reference therefore does not address the issue of how to ensure adequate performance for a very thin client platform. Moreover, it is difficult to determine, how, if at all, the system can perform real-time word recognition, and there is no meaningful description of how to integrate the system with a natural language processor.

The second of these references—U.S. Pat. No. 5,960, 399—Client/Server Speech Processor/Recognizer (assigned to GTE) describes the implementation of a HMM-based distributed speech recognition system. This reference is not instructive in many respects, however, including how to optimize acoustic feature extraction for a variety of client platforms, such as by performing a partial word recognition process where appropriate. Most importantly, there is only a description of a primitive server-based recognizer that only recognizes the user's speech and simply returns certain keywords such as the user's name and travel destination to fill out a dedicated form on the user's machine. Also, the $\,^{55}$ streaming of the acoustic parameters does not appear to be implemented in real-time as it can only take place after silence is detected. Finally, while the reference mentions the possible use of natural language processing (column 9) there is no explanation of how such function might be implemented in a real-time fashion to provide an interactive feel for the user

SUMMARY OF THE INVENTION

an improved system and method for overcoming the limitations of the prior art noted above;

A primary object of the present invention is to provide a word and phrase recognition system that is flexibly and optimally distributed across a client/platform computing architecture, so that improved accuracy, speed and uniformity can be achieved for a wide group of users;

A further object of the present invention is to provide a speech recognition system that efficiently integrates a distributed word recognition system with a natural language processing system, so that both individual words and entire speech utterances can be quickly and accurately recognized in any number of possible languages;

A related object of the present invention is to provide an efficient query response system so that an extremely accurate, real-time set of appropriate answers can be given in response to speech-based queries;

Yet another object of the present invention is to provide an interactive, real-time instructional/learning system that is distributed across a client/server architecture, and permits a real-time question/answer session with an interactive character:

A related object of the present invention is to implement such interactive character with an articulated response capability so that the user experiences a human-like interaction;

Still a further object of the present invention is to provide an INTERNET website with speech processing capability so that voice based data and commands can be used to interact with such site, thus enabling voice-based e-commerce and e-support services to be easily scaleable;

Another object is to implement a distributed speech recognition system that utilizes environmental variables as part of the recognition process to improve accuracy and

A further object is to provide a scaleable query/response database system, to support any number of query topics and users as needed for a particular application and instantaneous demand;

Yet another object of the present invention is to provide a query recognition system that employs a two-step approach, including a relatively rapid first step to narrow down the list of potential responses to a smaller candidate set, and a second more computationally intensive second step to identify the best choice to be returned in response to the query from the candidate set;

A further object of the present invention is to provide a natural language processing system that facilitates query recognition by extracting lexical components of speech utterances, which components can be used for rapidly identifying a candidate set of potential responses appropriate for such speech utterances;

Another related object of the present invention is to provide a natural language processing system that facilitates query recognition by comparing lexical components of speech utterances with a candidate set of potential response to provide an extremely accurate best response to such query.

One general aspect of the present invention, therefore, relates to a natural language query system (NLQS) that offers a fully interactive method for answering user's questions over a distributed network such as the INTERNET or a local intranet. This interactive system when implemented over the worldwide web (WWW) services of the INTER-NET functions so that a client or user can ask a question in An object of the present invention, therefore, is to provide 65 a natural language such as English, French, German or Spanish and receive the appropriate answer at his or her personal computer also in his or her native natural language.

matches the user's question. The metric that is used to determine the best possible stored question is the number of noun phrases. The stored answer that is paired to the best-stored question is selected as the one that answers the user's question. The ID tag of the question is then passed to the DBProcess.

The system is distributed and consists of a set of integrated software modules at the client's machine and another set of integrated software programs resident on a server or set of servers. The client-side software program is comprised of a speech recognition program, an agent and its control program, and a communication program. The server-side program is comprised of a communication program, a natural language engine (NLE), a database processor (DBProcess), an interface program for interfacing the DBProcess with the NLE, and a SQL database. In addition, the client's machine is equipped with a microphone and a speaker. Processing of the speech utterance is divided between the client and server side so as to optimize processing and transmission latencies, and so as to provide support for even very thin client platforms.

This DBProcess returns the answer which is stored in a file. A communication link is again established to send the answer back to the client in compressed form. The answer once received by the client is decompressed and articulated to the user by the text-to-speech engine. Thus, the invention can be used in any number of different applications involving interactive learning systems, INTERNET related commerce sites, INTERNET search engines, etc.

In the context of an interactive learning application, the system is specifically used to provide a single-best answer to a user's question. The question that is asked at the client's machine is articulated by the speaker and captured by a microphone that is built in as in the case of a notebook computer or is supplied as a standard peripheral attachment. Once the question is captured, the question is processed partially by NLQS client-side software resident in the client's machine. The output of this partial processing is a set of speech vectors that are transported to the server via the INTERNET to complete the recognition of the user's questions. This recognized speech is then converted to text at the server.

Computer-assisted instruction environments often require the assistance of mentors or live teachers to answer questions from students. This assistance often takes the form of organizing a separate pre-arranged forum or meeting time that is set aside for chat sessions or live call-in sessions so that at a scheduled time answers to questions may be provided. Because of the time immediacy and the on-demand or asynchronous nature of on-line training where a student may log on and take instruction at any time and at any location, it is important that answers to questions be provided in a timely and cost-effective manner so that the user or student can derive the maximum benefit from the material presented.

After the user's question is decoded by the speech recognition engine (SRE) located at the server, the question is 30 converted to a structured query language (SQL) query. This query is then simultaneously presented to a software process within the server called DBProcess for preliminary processing and to a Natural Language Engine (NLE) module for extracting the noun phrases (NP) of the user's question. 35 During the process of extracting the noun phrase within the NLE, the tokens of the users' question are tagged. The tagged tokens are then grouped so that the NP list can be determined. This information is stored and sent to the DBProcess process.

This invention addresses the above issues. It provides the user or student with answers to questions that are normally channeled to a live teacher or mentor. This invention provides a single-best answer to questions asked by the student. The student asks the question in his or her own voice in the language of choice. The speech is recognized and the answer to the question is found using a number of technologies including distributed speech recognition, full-text search database processing, natural language processing and textto-speech technologies. The answer is presented to the user, as in the case of a live teacher, in an articulated manner by an agent that mimics the mentor or teacher, and in the language of choice—English, French, German, Japanese or other natural spoken language. The user can choose the agent's gender as well as several speech parameters such as pitch, volume and speed of the character's voice.

In the DBProcess, the SQL query is fully customized using the NP extracted from the user's question and other environment variables that are relevant to the application. For example, in a training application, the user's selection of course, chapter and or section would constitute the environ- 45 ment variables. The SQL query is constructed using the extended SQL Full-Text predicates—CONTAINS, FREETEXT, NEAR, AND. The SQL query is next sent to the Full-Text search engine within the SQL database, where a Full-Text search procedure is initiated. The result of this 50 search procedure is recordset of answers. This recordset contains stored questions that are similar linguistically to the user's question. Each of these stored questions has a paired answer stored in a separate text file, whose path is stored in a table of the database.

Other applications that benefit from NLQS are e-commerce applications. In this application, the user's query for a price of a book, compact disk or for the availability of any item that is to be purchased can be retrieved without the need to pick through various lists on successive web pages. Instead, the answer is provided directly to the user without any additional user input.

The entire recordset of returned stored answers is then returned to the NLE engine in the form of an array. Each stored question of the array is then linguistically processed sequentially one by one. This linguistic processing constitutes the second step of a 2-step algorithm to determine the 60 single best answer to the user's question. This second step proceeds as follows: for each stored question that is returned in the recordset, a NP of the stored question is compared with the NP of the user's question. After all stored questions of the array are compared with the user's question, the stored question that yields the maximum match with the user's question is selected as the best possible stored question that

Similarly, it is envisioned that this system can be used to provide answers to frequently-asked questions (FAQs), and as a diagnostic service tool for e-support. These questions are typical of a give web site and are provided to help the user find information related to a payment procedure or the specifications of, or problems experienced with a product/ service. In all of these applications, the NLQS architecture can be applied.

A number of inventive methods associated with these architectures are also beneficially used in a variety of INTERNET related applications.

Although the inventions are described below in a set of preferred embodiments, it will be apparent to those skilled in the art the present inventions could be beneficially used in many environments where it is necessary to implement fast, accurate speech recognition, and/or to provide a human-like dialog capability to an intelligent system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a preferred embodiment of a natural language query system (NLQS) of the present invention, which is distributed across a client/server computing architecture, and can be used as an interactive learning system, an e-commerce system, an c-support system, and the like;

FIGS. 2A-2C is a block diagram of a preferred embodiment of a client side system, including speech capturing modules, partial speech processing modules, encoding modules, transmission modules, agent control modules, and answer/voice feedback modules that can be used in the aforementioned NLQS;

FIG. 2D is a block diagram of a preferred embodiment of 15 a set of initialization routines and procedures used for the client side system of FIGS. 2A-2C;

FIG. 3 is a block diagram of a preferred embodiment of a set of routines and procedures used for handling an iterated set of speech utterances on the client side system of FIG. 2, $_{20}$ transmitting speech data for such utterances to a remote server, and receiving appropriate responses back from such server:

FIG. 4 is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for 25 un-initializing the client side system of FIGS. 2A–2C;

FIG. 4A is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a distributed component of a speech recognition module for the server side system of FIG. 5;

FIG. 4B is a block diagram of a preferred set of routines and procedures used for implementing an SQL query builder for the server side system of FIG. 5;

FIG. 4C is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a 35 database control process module for the server side system of FIG. 5;

FIG. 4D is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a natural language engine that provides query formulation support, a query response module, and an interface to the database control process module for the server side system of FIG. 5;

FIG. 5 is a block diagram of a preferred embodiment of a server side system, including a speech recognition module to complete processing of the speech utterances, environmental and grammar control modules, query formulation modules, a natural language engine, a database control module, and a query response module that can be used in the aforementioned NLQS;

FIG. 6 illustrates the organization of a full-text database used as part of server side system shown in FIG. 5;

FIG. 7A illustrates the organization of a full-text database course table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention:

FIG. 7B illustrates the organization of a full-text database chapter table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present 60

FIG. 7C describes the fields used in a chapter table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention;

FIG. 7D describes the fields used in a section table used 65 as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention;

10

FIG. 8 is a flow diagram of a first set of operations performed by a preferred embodiment of a natural language engine on a speech utterance including Tokenization, Tagging and Grouping;

FIG. 9 is a flow diagram of the operations performed by a preferred embodiment of a natural language engine on a speech utterance including stemming and Lexical Analysis

FIG. 10 is a block diagram of a preferred embodiment of a SQL database search and support system for the present

FIGS. 11A-11C are flow diagrams illustrating steps performed in a preferred two step process implemented for query recognition by the NLQS of FIG. 2;

FIG. 12 is an illustration of another embodiment of the present invention implemented as part of a Web-based speech based learning/training System;

FIGS. 13-17 are illustrations of another embodiment of the present invention implemented as part of a Web-based e-commerce system;

FIG. 18 is an illustration of another embodiment of the present invention implemented as part of a voice-based Help Page for an E-Commerce Web Site.

DETAILED DESCRIPTION OF THE INVENTION

Overview

As alluded to above, the present inventions allow a user to ask a question in a natural language such as English, 30 French, German, Spanish or Japanese at a client computing system (which can be as simple as a personal digital assistant or cell-phone, or as sophisticated as a high end desktop PC) and receive an appropriate answer from a remote server also in his or her native natural language. As such, the embodiment of the invention shown in FIG. 1 is beneficially used in what can be generally described as a Natural Language Query System (NLQS) 100, which is configured to interact on a real-time basis to give a human-like dialog capability/ experience for e-commerce, e-support, and e-learning appli-

The processing for NLQS 100 is generally distributed across a client side system 150, a data link 160, and a server-side system 180. These components are well known in the art, and in a preferred embodiment include a personal computer system 150, an INTERNET connection 160A, 160B, and a larger scale computing system 180. It will be understood by those skilled in the art that these are merely exemplary components, and that the present invention is by no means limited to any particular implementation or com-50 bination of such systems. For example, client-side system 150 could also be implemented as a computer peripheral, a PDA, as part of a cell-phone, as part of an INTERNETadapted appliance, an INTERNET linked kiosk, etc. Similarly, while an INTERNET connection is depicted for data link 160A, it is apparent that any channel that is suitable for carrying data between client system 150 and server system 180 will suffice, including a wireless link, an RF link, an IR link, a LAN, and the like. Finally, it will be further appreciated that server system 180 may be a single, largescale system, or a collection of smaller systems interlinked to support a number of potential network users.

Initially speech input is provided in the form of a question or query articulated by the speaker at the client's machine or personal accessory as a speech utterance. This speech utterance is captured and partially processed by NLQS clientside software 155 resident in the client's machine. To facilitate and enhance the human-like aspects of the

11

interaction, the question is presented in the presence of an animated character 157 visible to the user who assists the user as a personal information retriever/agent. The agent can also interact with the user using both visible text output on a monitor/display (not shown) and/or in audible form using a text to speech engine 159. The output of the partial processing done by SRE 155 is a set of speech vectors that are transmitted over communication channel 160 that links the user's machine or personal accessory to a server or servers via the INTERNET or a wireless gateway that is linked to the INTERNET as explained above. At server 180, the partially processed speech signal data is handled by a server-side SRE 182, which then outputs recognized speech text corresponding to the user's question. Based on this user question related text, a text-to-query converter 184 formulates a suitable query that is used as input to a database processor 186. Based on the query, database processor 186 then locates and retrieves an appropriate answer using a customized SQL query from database 188. A Natural Language Engine 190 facilitates structuring the query to database 188. After a matching answer to the user's question is 20 found, the former is transmitted in text form across data link 160B, where it is converted into speech by text to speech engine 159, and thus expressed as oral feedback by animated character agent 157.

Because the speech processing is broken up in this 25 fashion, it is possible to achieve real-time, interactive, human-like dialog consisting of a large, controllable set of questions/answers. The assistance of the animated agent 157 further enhances the experience, making it more natural and comfortable for even novice users. To make the speech 30 recognition process more reliable, context-specific grammars and dictionaries are used, as well as natural language processing routines at NLE 190, to analyze user questions lexically. While context-specific processing of speech data is known in the art (see e.g., U.S. Pat. Nos. 5,960,394, 5,867, 35 817, 5,758,322 and 5,384,892 incorporated by reference herein) the present inventors are unaware of any such implementation as embodied in the present inventions. The text of the user's question is compared against text of other questions to identify the question posed by the user by DB 40 processor/engine (DBE) 186. By optimizing the interaction and relationship of the SR engines 155 and 182, the NLP routines 190, and the dictionaries and grammars, an extremely fast and accurate match can be made, so that a unique and responsive answer can be provided to the user. 45

On the server side 180, interleaved processing further accelerates the speech recognition process. In simplified terms, the query is presented simultaneously both to NLE 190 after the query is formulated, as well as to DBE 186. NLE 190 and SRE 182 perform complementary functions in 50 the overall recognition process. In general, SRE 182 is primarily responsible for determining the identity of the words articulated by the user, while NLE 190 is responsible for the linguistic morphological analysis of both the user's query and the search results returned after the database 55 query.

After the user's query is analyzed by NLE 190 some parameters are extracted and sent to the DBProcess. Additional statistics are stored in an array for the 2nd step of processing. During the 2nd step of 2-step algorithm, the recordset of preliminary search results are sent to the NLE 160 for processing. At the end of this 2nd step, the single question that matches the user's query is sent to the DBProcess where further processing yields the paired answer that is paired with the single best stored question.

Thus, the present invention uses a form of natural language processing (NLP) to achieve optimal performance in

a speech based web application system. While NLP is known in the art, prior efforts in Natural Language Processing (NLP) work nonetheless have not been well integrated with Speech Recognition (SR) technologies to achieve reasonable results in a web-based application environment. In speech recognition, the result is typically a lattice of possible recognized words each with some probability of fit with the speech recognizer. As described before, the input to a typical NLP system is typically a large linguistic unit. The NLP system is then charged with the parsing, understanding and indexing of this large linguistic unit or set of transcribed utterances. The result of this NLP process is to understand lexically or morphologically the entire linguistic unit as opposed to word recognition. Put another way, the linguistic unit or sentence of connected words output by the SRE has to be understood lexically, as opposed to just being "recog-

12

Filed 09/08/2008

As indicated earlier, although speech recognition technology has been available for several years, the technical requirements for the NLQS invention have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily. In realizing that even with the best of conditions, it might be not be possible to achieve the perfect 100% speech recognition accuracy that is required, the present invention employs an algorithm that balances the potential risk of the speech recognition process with the requirements of the natural language processing so that even in cases where perfect speech recognition accuracy is not achieved for each word in the query, the entire query itself is nonetheless recognized with sufficient accuracy.

This recognition accuracy is achieved even while meeting very stringent user constraints, such as short latency periods of 3 to 5 seconds (ideally—ignoring transmission latencies which can vary) for responding to a speech-based query, and for a potential set of 100-250 query questions. This quick response time gives the overall appearance and experience of a real-time discourse that is more natural and pleasant from the user's perspective. Of course, non-real time applications, such as translation services for example, can also benefit from the present teachings as well, since a centralized set of HMMs, grammars, dictionaries, etc., are

General Aspects of Speech Recognition Used in the Present

General background information on speech recognition can be found in the prior art references discussed above and incorporated by reference herein. Nonetheless, a discussion of some particular exemplary forms of speech recognition structures and techniques that are well-suited for NLQS 100 is provided next to better illustrate some of the characteristics, qualities and features of the present inven-

Speech recognition technology is typically of two types speaker independent and speaker dependent. In speakerdependent speech recognition technology, each user has a voice file in which a sample of each potentially recognized word is stored. Speaker-dependent speech recognition systems typically have large vocabularies and dictionaries making them suitable for applications as dictation and text transcribing. It follows also that the memory and processor resource requirements for the speaker-dependent can be and are typically large and intensive.

Conversely speaker-independent speech recognition technology allows a large group of users to use a single vocabulary file. It follows then that the degree of accuracy that can

13

be achieved is a function of the size and complexity of the grammars and dictionaries that can be supported for a given language. Given the context of applications for which NLQS, the use of small grammars and dictionaries allow speaker independent speech recognition technology to be implemented in NLQS.

The key issues or requirements for either type—speaker-independent or speaker-dependent, are accuracy and speed. As the size of the user dictionaries increase, the speech recognition accuracy metric—word error rate (WER) and 10 the speed of recognition decreases. This is so because the search time increases and the pronunciation match becomes more complex as the size of the dictionary increases.

The basis of the NLQS speech recognition system is a series of Hidden Markov Models (HMM), which, as alluded to earlier, are mathematical models used to characterize any time varying signal. Because parts of speech are considered to be based on an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The 20 Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, 25 according to an associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O, is generated with probability density B_i(O_t). It is only the outcome, not the state which is visible to an external 30 observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.

In isolated speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word can each be described by a Markov model as follows: 35

$$O=o_1, o_2, \dots o_T \tag{1-1}$$

where o, is a speech vector observed at time t. The isolated word recognition then is to compute:

$$\arg\max\{P(w_i|O)\}\tag{1-2}$$

By using Bayes' Rule,

$${P(w_i|O)}=[P(O|w_i)P(w_i)]/P(O)$$
 (1-3)

In the general case, the Markov model when applied to speech also assumes a finite state machine which changes state once every time unit and each time that a state j is entered, a speech vector o_i is generated from the probability density $b_j(o_i)$. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij} .

For a state sequence X, the joint probability that 0 is generated by the model M moving through a state sequence X is the product of the transition probabilities and the output 55 probabilities. Only the observation sequence is known—the state sequence is hidden as mentioned before.

Given that X is unknown, the required likelihood is computed by summing over all possible state sequences X=x(1), x(2), x(3), . . . x(T), that is

$$P(O|M) = \sum \big\{ a_{x(0) \ x(1)} \Pi b(x) \ (o_i) a_{x(i) \ x(i+1)} \big\}$$

Given a set of models M_i , corresponding to words w_i equation 1-2 is solved by using 1-3 and also by assuming that:

$$P(O|w_i)=P(O|M_i)$$

14

All of this assumes that the parameters $\{a_{ij}\}$ and $\{b_i(o_t)\}$ are known for each model Mi. This can be done, as explained earlier, by using a set of training examples corresponding to a particular model. Thereafter, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure. So if a sufficient number of representative examples of each word are collected, then a HMM can be constructed which simply models all of the many sources of variability inherent in real speech. This training is well-known in the art, so it is not described at length herein, except to note that the distributed architecture of the present invention enhances the quality of HMMs, since they are derived and constituted at the server side, rather than the client side. In this way, appropriate samples from users of different geographical areas can be easily compiled and analyzed to optimize the possible variations expected to be seen across a particular language to be recognized. Uniformity of the speech recognition process is also wellmaintained, and error diagnostics are simplified, since each prospective user is using the same set of HMMs during the recognition process.

To determine the parameters of a HMM from a set of training samples, the first step typically is to make a rough guess as to what they might be. Then a refinement is done using the Baum-Welch estimation formulae. By these formulae, the maximum likelihood estimates of μ_j (where μ_j is mean vector and Σ_i is covariance matrix) is:

$$\mu_{j} \! = \! \boldsymbol{\Sigma}^{T}_{t=1} L_{j}(t) o_{t} \! / \! \big[\boldsymbol{\Sigma}^{T}_{t=1} L_{j}(t) o_{t} \big]$$

A forward-backward algorithm is next used to calculate the probability of state occupation $L_j(t)$. If the forward probability $\alpha_j(t)$ for some model M with N states is defined as:

$$\alpha_i(t)=P(o_1,\ldots,o_t,x(t)=j|M)$$

This probability can be calculated using the recursion:

$$\alpha_{i}(t) = [\Sigma^{N-1}_{i=2}\alpha(t-1)a_{ij}]b_{i}(o_{t})$$

Similarly the backward probability can be computed using the recursion:

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij} b_j(o_{t+1})(t+1)$$

Realizing that the forward probability is a joint probability and the backward probability is a conditional probability, the probability of state occupation is the product of the two probabilities:

$$\alpha j(t)\beta_j(t){=}P(O{,}x(t){=}j\big|M)$$

Hence the probability of being in state j at a time t is:

$$L_i(t)=1/P[\alpha_i(t)\beta_i(t)]$$

where P=P(O|M)

To generalize the above for continuous speech recognition, we assume the maximum likelihood state sequence where the summation is replaced by a maximum operation. Thus for a given model M, let ϕj (t) represent the maximum likelihood of observing speech vectors o_1 to o_r and being used in state j at time t:

$$\phi_j(t) = \max\{\phi_j(t)(t-1)\alpha_{ij}\}\beta_j(o_i)$$

Expressing this logarithmically to avoid underflow, this likelihood becomes:

$$\psi_i(t) = \max\{\psi_i(t-1) + \log(\alpha_{ij})\} + \log(b_i(o_i))$$

15

This is also known as the Viterbi algorithm. It can be visualized as finding the best path through a matrix where the vertical dimension represents the states of the HMM and horizontal dimension represents frames of speech i.e. time. To complete the extension to connected speech recognition, it is further assumed that each HMM representing the underlying sequence is connected. Thus the training data for continuous speech recognition should consist of connected utterances; however, the boundaries between words do not have to be known.

To improve computational speed/efficiency, the Viterbi algorithm is sometimes extended to achieve convergence by using what is known as a Token Passing Model. The token passing model represents a partial match between the observation sequence o₁ to o₂ and a particular model, subject to the constraint that the model is in state j at time t. This token passing model can be extended easily to connected speech environments as well if we allow the sequence of HMMs to be defined as a finite state network. A composite network that includes both phoneme-based HMMs and complete 20 words can be constructed so that a single-best word can be recognized to form connected speech using word N-best extraction from the lattice of possibilities. This composite form of HMM-based connected speech recognizer is the basis of the NLQS speech recognizer module. Nonetheless, 25 the present invention is not limited as such to such specific forms of speech recognizers, and can employ other techniques for speech recognition if they are otherwise compatible with the present architecture and meet necessary performance criteria for accuracy and speed to provide a 30 real-time dialog experience for users.

The representation of speech for the present invention's HMM-based speech recognition system assumes that speech is essentially either a quasi-periodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced 35 sounds). It may be modeled as two sources—one a impulse train generator with pitch period P and a random noise generator which is controlled by a voice/unvoiced switch. The output of the switch is then fed into a gain function estimated from the speech signal and scaled to feed a digital 40 filter H(z) controlled by the vocal tract parameter characteristics of the speech being produced. All of the parameters for this model—the voiced/unvoiced switching, the pitch period for voiced sounds, the gain parameter for the speech signal and the coefficient of the digital filter, vary slowly with time. In extracting the acoustic parameters from the user's speech input so that it can evaluated in light of a set of HMMs, cepstral analysis is typically used to separate the vocal tract information from the excitation information. The cepstrum of a signal is computed by taking the Fourier (or 50 recognition system, the entire SR engine runs on a single similar) transform of the log spectrum. The principal advantage of extracting cepstral coefficients is that they are de-correlated and the diagonal covariances can be used with HMMs. Since the human ear resolves frequencies nonlinearly across the audio spectrum, it has been shown that a 55 front-end that operates in a similar non-linear way improves speech recognition performance.

Accordingly, instead of a typical linear prediction-based analysis, the front-end of the NLQS speech recognition engine implements a simple, fast Fourier transform based filter bank designed to give approximately equal resolution on the Mel-scale. To implement this filter bank, a window of speech data (for a particular time frame) is transformed using a software based Fourier transform and the magnitude taken. Each FFT magnitude is then multiplied by the corresponding filter gain and the results accumulated. The cepstral coefficients that are derived from this filter-bank

16

analysis at the front end are calculated during a first partial processing phase of the speech signal by using a Discrete Cosine Transform of the log filter bank amplitudes. These cepstral coefficients are called Mel-Frequency Cepstral Coefficients (MFCC) and they represent some of the speech parameters transferred from the client side to characterize the acoustic features of the user's speech signal. These parameters are chosen for a number of reasons, including the fact that they can be quickly and consistently derived even 10 across systems of disparate capabilities (i.e., for everything from a low power PDA to a high powered desktop system), they give good discrimination, they lend themselves to a number of useful recognition related manipulations, and they are relatively small and compact in size so that they can be transported rapidly across even a relatively narrow band link. Thus, these parameters represent the least amount of information that can be used by a subsequent server side system to adequately and quickly complete the recognition process.

To augment the speech parameters an energy term in the form of the logarithm of the signal energy is added. Accordingly, RMS energy is added to the 12 MFCC's to make 13 coefficients. These coefficients together make up the partially processed speech data transmitted in compressed form from the user's client system to the remote server side.

The performance of the present speech recognition system is enhanced significantly by computing and adding time derivatives to the basic static MFCC parameters at the server side. These two other sets of coefficients—the delta and acceleration coefficients representing change in each of the 13 values from frame to frame (actually measured across several frames), are computed during a second partial speech signal processing phase to complete the initial processing of the speech signal, and are added to the original set of coefficients after the latter are received. These MFCCs together with the delta and acceleration coefficients constitute the observation vector O_t mentioned above that is used for determining the appropriate HMM for the speech data.

The delta and acceleration coefficients are computed using the following regression formula:

$$d_{i} = \Sigma^{\theta}_{\theta=1} [c_{i+\theta} - c_{i-\theta}]/2 \Sigma^{\theta}_{\theta=1} \theta^{2}$$

where d, is a delta coefficient at time t computed in terms of the corresponding static coefficients:

$$d_i = [c_{i+\theta} - c_{i-\theta}]/2\theta$$

In a typical stand-alone implementation of a speech client. In other words, both the first and second partial processing phases above are executed by the same DSP (or microprocessor) running a ROM or software code routine at the client's computing machine.

In contrast, because of several considerations, specifically—cost, technical performance, and client hardware uniformity, the present NLQS system uses a partitioned or distributed approach. While some processing occurs on the client side, the main speech recognition engine runs on a centrally located server or number of servers. More specifically, as noted earlier, capture of the speech signals, MFCC vector extraction and compression are implemented on the client's machine during a first partial processing phase. The routine is thus streamlined and simple enough to be implemented within a browser program (as a plug in module, or a downloadable applet for example) for maximum ease of use and utility. Accordingly, even very "thin"

17

client platforms can be supported, which enables the use of the present system across a greater number of potential sites. The primary MFCCs are then transmitted to the server over the channel, which, for example, can include a dial-up INTERNET connection, a LAN connection, a wireless connection and the like. After decompression, the delta and acceleration coefficients are computed at the server to complete the initial speech processing phase, and the resulting observation vectors O, are also determined.

General Aspects of Speech Recognition Engine

The speech recognition engine is also located on the server, and is based on a HTK-based recognition network compiled from a word-level network, a dictionary and a set of HMMs. The recognition network consists of a set of nodes connected by arcs. Each node is either a HMM model instance or a word end. Each model node is itself a network consisting of states connected by arcs. Thus when fully compiled, a speech recognition network consists of HMM states connected by transitions. For an unknown input utterance with T frames, every path from the start node to the exit 20 node of the network passes through T HMM states. Each of these paths has log probability which is computed by summing the log probability of each individual transition in the path and the log probability of each emitting state generating the corresponding observation. The function of 25 the Viterbi decoder is find those paths through the network which have the highest log probability. This is found using the Token Passing algorithm. In a network that has many nodes, the computation time is reduced by only allowing propagation of those tokens which will have some chance of 30 becoming winners. This process is called pruning. Natural Language Processor

In a typical natural language interface to a database, the user enters a question in his/her natural language, for example, English. The system parses it and translates it to a 35 query language expression. The system then uses the query language expression to process the query and if the search is successful, a recordset representing the results is displayed in English either formatted as raw text or in a graphical form. For a natural language interface to work well involves a 40 number of technical requirements.

For example, it needs to be robust—in the sentence 'What's the departments turnover' it needs to decide that the word whats=what's=what is. And it also has to determine that departments=department's. In addition to being robust, the natural language interface has to distinguish between the several possible forms of ambiguity that may exist in the natural language—lexical, structural, reference and ellipsis ambiguity. All of these requirements, in addition to the general ability to perform basic linguistic morphological 50 operations of tokenization, tagging and grouping, are implemented within the present invention.

Tokenization is implemented by a text analyzer which treats the text as a series of tokens or useful meaningful units that are larger than individual characters, but smaller than 55 the search criteria. phrases and sentences. These include words, separable parts of words, and punctuation. Each token is associated with an offset and a length. The first phase of tokenization is the process of segmentation which extracts the individual tokens from the input text and keeps track of the offset where each token originated in the input text. The tokenizer output lists the offset and category for each token. In the next phase of the text analysis, the tagger uses a built-in morphological analyzer to look up each word/token in a phrase or sentence and internally lists all parts of speech. The output is the input 65 string with each token tagged with a parts of speech notation. Finally the grouper which functions as a phrase extrac18

tor or phrase analyzer, determines which groups of words form phrases. These three operations which are the foundations for any modem linguistic processing schemes, are fully implemented in optimized algorithms for determining the single-best possible answer to the user's question. SQL Database and Full-Text Query

Another key component of present system is a SQLdatabase. This database is used to store text, specifically the answer-question pairs are stored in full-text tables of the 10 database. Additionally, the full-text search capability of the database allows full-text searches to be carried out.

While a large portion of all digitally stored information is in the form of unstructured data, primarily text, it is now possible to store this textual data in traditional database systems in character-based columns such as varchar and text. In order to effectively retrieve textual data from the database, techniques have to be implemented to issue queries against textual data and to retrieve the answers in a meaningful way where it provides the answers as in the case of the NLQS system.

There are two major types of textual searches: Property— This search technology first applies filters to documents in order to extract properties such as author, subject, type, word count, printed page count, and time last written, and then issues searches against those properties; Full-text—this search technology first creates indexes of all non-noise words in the documents, and then uses these indexes to support linguistic searches and proximity searches.

Two additional technologies are also implemented in this particular RDBMs: SQL Server also have been integrated: A Search service—a full-text indexing and search service that is called both index engine and search, and a parser that accepts full-text SQL extensions and maps them into a form that can be processed by the search engine.

The four major aspects involved in implementing full-text retrieval of plain-text data from a full-text-capable database are: Managing the definition of the tables and columns that are registered for full-text searches; Indexing the data in registered columns—the indexing process scans the character streams, determines the word boundaries (this is called word breaking), removes all noise words (this also is called stop words), and then populates a full-text index with the remaining words; Issuing queries against registered columns for populated full-text indexes; Ensuring that subsequent changes to the data in registered columns gets propagated to the index engine to keep the full-text indexes synchronized.

The underlying design principle for the indexing, querying, and synchronizing processes is the presence of a full-text unique key column (or single-column primary key) on all tables registered for full-text searches. The full-text index contains an entry for the non-noise words in each row together with the value of the key column for each row.

When processing a full-text search, the search engine returns to the database the key values of the rows that match

The full-text administration process starts by designating a table and its columns of interest for full-text search. Customized NLQS stored procedures are used first to register tables and columns as eligible for full-text search. After that, a separate request by means of a stored procedure is issued to populate the full-text indexes. The result is that the underlying index engine gets invoked and asynchronous index population begins. Full-text indexing tracks which significant words are used and where they are located. For example, a full-text index might indicate that the word "NLQS" is found at word number 423 and word number 982 in the Abstract column of the DevTools table for the row

19

associated with a ProductID of 6. This index structure supports an efficient search for all items containing indexed words as well as advanced search operations, such as phrase searches and proximity searches. (An example of a phrase search is looking for "white elephant," where "while" is followed by "elephant". An example of a proximity search is looking for "big" and "house" where "big" occurs near "house".) To prevent the full-text index from becoming bloated, noise words such as "a," "and," and "the" are ignored.

Extensions to the Transact-SQL language are used to construct full-text queries. The two key predicates that are used in the NLQS are CONTAINS and FREETEXT.

The CONTAINS predicate is used to determine whether or not values in full-text registered columns contain certain 15 words and phrases. Specifically, this predicate is used to search for:

A word or phrase.

The prefix of a word or phrase.

A word or phrase that is near another.

A word that is an inflectional form of another (for example, "drive" is the inflectional stem of "drives," "drove," "driving" and "driven").

A set of words or phrases, each of which is assigned a 25 different weighting.

The relational engine within SQL Server recognizes the CONTAINS and FREETEXT predicates and performs some minimal syntax and semantic checking, such as ensuring that the column referenced in the predicate has been registered for full-text searches. During query execution, a full-text predicate and other relevant information are passed to the full-text search component. After further syntax and semantic validation, the search engine is invoked and returns the set of unique key values identifying those rows in the 35 table that satisfy the full-text search condition. In addition to the FREETEXT and CONTAINS, other predicates such as AND, LIKE, NEAR are combined to create the customized NLOS SOL construct.

Full-Text Query Architecture of the SQL Database

The full-text query architecture is comprised of the following several components—Full-Text Query component, the SQL Server Relational Engine, the Full-Text provider and the Search Engine.

The Full-Text Query component of the SQL database 45 accept a full-text predicate or rowset-valued function from the SQL Server; transform parts of the predicate into an internal format, and sends it to Search Service, which returns the matches in a rowset. The rowset is then sent back to SQL Server. SQL Server uses this information to create the 50 resultset that is then returned to the submitter of the query.

The SQL Server Relational Engine accepts the CON-TAINS and FREETEXT predicates as well as the CONTAINSTABLE() and FREETEXTTABLE() rowsetvalued functions. During parse time, this code checks for 55 conditions such as attempting to query a column that has not been registered for full-text search. If valid, then at run time, the ft_search_condition and context information is sent to the full-text provider. Eventually, the full-text provider returns a rowset to SQL Server, which is used in any joins 60 (specified or implied) in the original query. The Full-Text Provider parses and validates the ft_search_condition, constructs the appropriate internal representation of the full-text search condition, and then passes it to the search engine. The result is returned to the relational engine by means of a 65 rowset of rows that satisfy ft_search_condition. Client Side System 150

20

The architecture of client-side system 150 of Natural Language Query System 100 is illustrated in greater detail in FIGS. 2A-2C. Referring to FIGS. 2A-2C, the three main processes effectuated by Client System 150 are illustrated as follows: Initialization process 200A consisting of SRE 201, Communication 202 and Microsoft (MS) Agent 203 routines; At FIG. 2B on an iterative process 200B consisting of two sub-routines: a) Receive User Speech 208—made up of SRE 204 and Communication 205; and b) Receive Answer from Server 207—made up of MS Speak Agent 206, Communication 209, Voice data file 210 and Text to Speech Engine 211. Finally, in FIG. 2C un-initialization process 200C is made up of three sub-routines: SRE 212, Communication 213, and MS Agent 214. Each of the above three processes are described in detail in the following paragraphs. It will be appreciated by those skilled in the art that the particular implementation for such processes and routines will vary from client platform to platform, so that in some environments such processes may be embodied in hardcoded routines executed by a dedicated DSP, while in others 20 they may be embodied as software routines executed by a shared host processor, and in still others a combination of the two may be used.

Initialization at Client System 150

The initialization of the Client System 150 is illustrated in FIG. 2D and is comprised generally of 3 separate initializing processes: client-side Speech Recognition Engine 220A, MS Agent 220B and Communication processes 220C. Initialization of Speech Recognition Engine 220A

Speech Recognition Engine 155 is initialized and configured using the routines shown in 220A. First, an SRE COM Library is initialized. Next, memory 220 is allocated to hold Source and Coder objects, are created by a routine 221. Loading of configuration file 221A from configuration data file 221B also takes place at the same time that the SRE Library is initialized. In configuration file 221B, the type of the input of Coder and the type of the output of the Coder are declared. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches. Accordingly, they are not discussed in detail herein. Next, Speech and Silence components of an utterance are calibrated using a routine 222, in a procedure that is also well-known in the art. To calibrate the speech and silence components, the user preferably articulates a sentence that is displayed in a text box on the screen. The SRE library then estimates the noise and other parameters required to find e silence and speech elements of future user utterances. Initialization of MS Agent 220B

The software code used to initialize and set up a MS Agent 220B is also illustrated in FIG. 2D. The MS Agent 220B routine is responsible for coordinating and handling the actions of the animated agent 157 (FIG.1). This initialization thus consists of the following steps:

- 1. Initialize COM library 223. This part of the code initializes the COM library, which is required to use ActiveX Controls, which controls are well-known in the art.
- 2. Create instance of Agent Server 224—this part of the code creates an instance of Agent ActiveX control.
- 3. Loading of MS Agent 225—this part of the code loads MS Agent character from a specified file 225A containing general parameter data for the Agent Character, such as the overall appearance, shape, size, etc.
- 4. Get Character Interface 226—this part of the code gets an appropriate interface for the specified character; for example, characters may have different control/ interaction capabilities that can be presented to the user.

21

- 5. Add Commands to Agent Character Option 227—this part of the code adds commands to an Agent Properties sheet, which sheet can be accessed by clicking on the icon that appears in the system tray, when the Agent character is loaded e.g., that the character can Speak, how he/she moves, TTS Properties, etc.
- 6. Show the Agent Character 228—this part of the code displays the Agent character on the screen so it can be seen by the user;
- 7. AgentNotifySink—to handle events. This part of the 10 code creates AgentNotifySink object 229, registers it at 230 and then gets the Agent Properties interface 231. The property sheet for the Agent character is assigned using routine 232
- 8. Do Character Animations 233—This part of the code 15 plays specified character animations to welcome the user to NLQS 100.

The above then constitutes the entire sequence required to initialize the MS Agent. As with the SRE routines, the MS Agent routines can be implemented in any suitable and 20 conventional fashion by those skilled in the art based on the present teachings. The particular structure, operation, etc. of such routines is not critical, and thus they are not discussed in detail herein.

In a preferred embodiment, the MS Agent is configured to 25 have an appearance and capabilities that are appropriate for the particular application. For instance, in a remote learning application, the agent has the visual form and mannerisms/ attitude/gestures of a college professor. Other visual props (blackboard, textbook, etc.) may be used by the agent and presented to the user to bring to mind the experience of being in an actual educational environment. The characteristics of the agent may be configured at the client side 150, and/or as part of code executed by a browser program (not shown) in response to configuration data and commands 35 from a particular web page. For example, a particular website offering medical services may prefer to use a visual image of a doctor. These and many other variations will be apparent to those skilled in the art for enhancing the humanlike, real-time dialog experience for users.

Initialization of Communication Link 160A

The initialization of Communication Link 160A is shown with reference to process 220C FIG. 2D. Referring to FIG. 2D, this initialization consists of the following code components: Open INTERNET Connection 234—this part of the 45 code opens an INTERNET Connection and sets the parameter for the connection. Then Set Callback Status routine 235 sets the callback status so as to inform the user of the status of connection. Finally Start New HTTP INTERNET Session 236 starts a new INTERNET session. The details of Com- 50 munications Link 160 and the set up process 220C are not critical, and will vary from platform to platform. Again, in some cases, users may use a low-speed dial-up connection, a dedicated high speed switched connection (T1 for example), an always-on xDSL connection, a wireless 55 connection, and the like.

Iterative Processing of Queries/Answers

As illustrated in FIG. 3, once initialization is complete, an iterative query/answer process is launched when the user presses the Start Button to initiate a query. Referring to FIG. 3, the iterative query/answer process consists of two main sub-processes implemented as routines on the client side system 150: Receive User Speech 240 and Receive User Answer 243. The Receive User Speech 240 routine receives speech from the user (or another audio input source), while the Receive User Answer 243 routine receives an answer to the user's question in the form of text from the server so that

22

it can be converted to speech for the user by text-to-speech engine 159. As used herein, the term "query" is referred to in the broadest sense to refer, to either a question, a command, or some form of input used as a control variable by the system. For example, a query may consist of a question directed to a particular topic, such as "what is a network" in the context of a remote learning application. In an e-commerce application a query might consist of a command to "list all books by Mark Twain" for example. Similarly, while the answer in a remote learning application consists of text that is rendered into audible form by the text to speech engine 159, it could also be returned as another form of multi-media information, such as a graphic image, a sound file, a video file, etc. depending on the requirements of the particular application. Again, given the present teachings concerning the necessary structure, operation, functions, performance, etc., of the client-side Receive User Speech 240 and Receiver User Answer 243 routines, one of ordinary skill in the art could implement such in a variety of

Receive User Speech—As illustrated in FIG. 3, the Receive User Speech routine 240 consists of a SRE 241 and a Communication 242 process, both implemented again as routines on the client side system 150 for receiving and partially processing the user's utterance. SRE routine 241 uses a coder 248 which is prepared so that a coder object receives speech data from a source object. Next the Start Source 249 routine is initiated. This part of the code initiates data retrieval using the source Object which will in turn be given to the Coder object. Next, MFCC vectors 250 are extracted from the Speech utterance continuously until silence is detected. As alluded to earlier, this represents the first phase of processing of the input speech signal, and in a preferred embodiment, it is intentionally restricted to merely computing the MFCC vectors for the reasons already expressed above. These vectors include the 12 cepstral coefficients and the RMS energy term, for a total of 13 separate numerical values for the partially processed speech signal.

In some environments, nonetheless, it is conceivable that the MFCC delta parameters and MFCC acceleration parameters can also be computed at client side system 150, depending on the computation resources available, the transmission bandwidth in data link 160A available to server side system 180, the speed of a transceiver used for carrying data in the data link, etc. These parameters can be determined automatically by client side system upon initializing SRE 155 (using some type of calibration routine to measure resources), or by direct user control, so that the partitioning of signal processing responsibilities can be optimized on a case-by-case basis. In some applications, too, server side system 180 may lack the appropriate resources or routines for completing the processing of the speech input signal. Therefore, for some applications, the allocation of signal processing responsibilities may be partitioned differently, to the point where in fact both phases of the speech signal processing may take place at client side system 150 so that the speech signal is completely—rather than partially processed and transmitted for conversion into a query at server side system 180.

Again in a preferred embodiment, to ensure reasonable accuracy and real-time performance from a query/response perspective, sufficient resources are made available in a client side system so that 100 frames per second of speech data can be partially processed and transmitted through link 160A. Since the least amount of information that is necessary to complete the speech recognition process (only 13

coefficients) is sent, the system achieves a real-time performance that is believed to be highly optimized, because other latencies (i.e., client-side computational latencies, packet formation latencies, transmission latencies) are minimized. It will be apparent that the principles of the present invention can be extended to other SR applications where some other methodology is used for breaking down the speech input signal by an SRE (i.e., non-MFCC based). The only criteria is that the SR processing be similarly dividable into multiple phases, and with the responsibility for different phases being handled on opposite sides of link 160A depending on overall system performance goals, requirements and the like. This functionality of the present invention can thus be achieved on a system-by-system basis, with an expected and typical amount of optimization being necessary for each particular 15 implementation.

Thus, the present invention achieves a response rate performance that is tailored in accordance with the amount of information that is computed, coded and transmitted by the client side system 150. So in applications where realtime performance is most critical, the least possible amount of extracted speech data is transmitted to reduce these latencies, and, in other applications, the amount of extracted speech data that is processed, coded and transmitted can be varied.

Communication—transmit communication module 242 is used to implement the transport of data from the client to the server over the data link 160A, which in a preferred embodiment is the INTERNET. As explained above, the data consists of encoded MFCC vectors that will be used at then server-side of the Speech Recognition engine to complete the speech recognition decoding. The sequence of the communication is as follows:

OpenHTTPRequest 251—this part of the code first converts MFCC vectors to a stream of bytes, and then processes 35 the bytes so that it is compatible with a protocol known as HTTP. This protocol is well-known in the art, and it is apparent that for other data links another suitable protocol would be used.

- encodes the MFCC vectors, so that they can be sent to the server via HTTP.
- 2. Send data 252—this part of the code sends MFCC vectors to the server using the INTERNET connection and the HTTP protocol.

Wait for the Server Response 253—this part of the code monitors the data link 160A a response from server side system 180 arrives. In summary, the MFCC parameters are extracted or observed on-the-fly from the input speech signal. They are then encoded to a HTTP byte stream and 50 sent in a streaming fashion to the server before the silence is detected-i.e. sent to server side system 180 before the utterance is complete. This aspect of the invention also facilitates a real-time behavior, since data can be transmitted and processed even while the user is still speaking.

Receive Answer from Server 243 is comprised of the following modules as shown in FIG. 3: MS Agent 244, Text-to-Speech Engine 245 and receive communication modules 246. All three modules interact to receive the answer from server side system 180. As illustrated in FIG. 3, the receive communication process consists of three separate processes implemented as a receive routine on client side system 150: a Receive the Best Answer 258 receives the best answer over data link 160B (the HTTP communication channel). The answer is de-compressed at 65 259 and then the answer is passed by code 260 to the MS Agent 244, where it is received by code portion 254. A

24

routine 255 then articulates the answer using text-to-speech engine 257. Of course, the text can also be displayed for additional feedback purposes on a monitor used with client side system 150. The text to speech engine uses a natural language voice data file 256 associated with it that is appropriate for the particular language application (i.e., English, French, German, Japanese, etc.). As explained earlier when the answer is something more than text, it can be treated as desired to provide responsive information to the user, such as with a graphics image, a sound, a video clip, etc.

Uninitialization

The un-initialization routines and processes are illustrated in FIG. 4. Three functional modules are used for un-initializing the primary components of the client side system 150; these include SRE 270, Communications 271 and MS Agent 272 un-initializing routines. To un-initialize SRE 220A, memory that was allocated in the initialization phase is de-allocated by code 273 and objects created during such initialization phase are deleted by code 274. Similarly, as illustrated in FIG. 4, to un-initialize Communications module 220C the INTERNET connection previously established with the server is closed by code portion 275 of the Communication Un-initialization routine 271. Next the INTERNET session created at the time of initialization is 25 also closed by routine 276. For the un-initialization of the MS Agent 220B, as illustrated in FIG. 4, MS Agent Uninitialization routine 272 first releases the Commands Interface 227 using routine 277. This releases the commands added to the property sheet during loading of the agent character by routine 225. Next the Character Interface initialized by routine 226 is released by routine 278 and the Agent is unloaded at 279. The Sink Object Interface is then also released 280 followed by the release of the Property Sheet Interface 281. The Agent Notify Sink 282 then un-registers the Agent and finally the Agent Interface 283 is released which releases all the resources allocated during initialization steps identified in FIG. 2D.

It will be appreciated by those skilled in the art that the particular implementation for such un-initialization pro-1. Encode MFCC Byte Stream 251—this part of the code 40 cesses and routines in FIG. 4 will vary from client platform to client platform, as for the other routines discussed above. The structure, operation, etc. of such routines are wellknown in the art, and they can be implemented using a number of fairly straightforward approaches without undue effort. Accordingly, they are not discussed in detail herein. Description of Server Side System 180 Introduction

> A high level flow diagram of the set of preferred processes implemented on server side system 180 of Natural Language Query System 100 is illustrated in FIGS. 11A through FIG. 11C. In a preferred embodiment, this process consists of a two step algorithm for completing the processing of the speech input signal, recognizing the meaning of the user's query, and retrieving an appropriate answer/response for such query.

> The 1st step as illustrated in FIG. 11A can be considered a high-speed first-cut pruning mechanism, and includes the following operations: after completing processing of the speech input signal, the user's query is recognized at step 1101, so that the text of the query is simultaneously sent to Natural Language Engine 190 (FIG. 1) at step 1107, and to DB Engine 186 (also FIG.1) at step 1102. By "recognized" in this context it is meant that the user's query is converted into a text string of distinct native language words through the HMM technique discussed earlier.

At NLE 190, the text string undergoes morphological linguistic processing at step 1108: the string is tokenized the

25

tags are tagged and the tagged tokens are grouped Next the noun phrases (NP) of the string are stored at 1109, and also copied and transferred for use by DB Engine 186 during a DB Process at step 1110. As illustrated in FIG. 11A, the string corresponding to the user's query which was sent to the DB Engine 186 at 1102, is used together with the NP received from NLE 190 to construct an SQL Query at step 1103. Next, the SQL query is executed at step 1104, and a record set of potential questions corresponding to the user's query are received as a result of a full-text search at 1105, 10 which are then sent back to NLE 190 in the form of an array at step 1106.

As can be seen from the above, this first step on the server side processing acts as an efficient and fast pruning mechanism so that the universe of potential "hits" corresponding to 15 the user's actual query is narrowed down very quickly to a manageable set of likely candidates in a very short period of time.

Referring to FIG. 11B, in contrast to the first step above, the 2^{nd} step can be considered as the more precise selection 20 portion of the recognition process. It begins with linguistic processing of each of the stored questions in the array returned by the full-text search process as possible candidates representing the user's query. Processing of these stored questions continues in NLE 190 as follows: each 25 question in the array of questions corresponding to the record set returned by the SQL full-text search undergoes morphological linguistic processing at step 1111: in this operation, a text string corresponding to the retrieved candidate question is tokenized, the tags are tagged and the 30 tagged tokens are grouped. Next, noun phrases of the string are computed and stored at step 1112. This process continues iteratively at point 1113, and the sequence of steps at 1118,1111, 1112, 1113 are repeated so that an NP for each retrieved candidate question is computed and stored. Once 35 an NP is computed for each of the retrieved candidate questions of the array, a comparison is made between each such retrieved candidate question and the user's query based on the magnitude of the NP value at step 1114. This process is also iterative in that steps 1114, 1115, 1116, 1119 are 40 180 repeated so that the comparison of the NP for each retrieved candidate question with that of the NP of the user's query is completed. When there are no more stored questions in the array to be processed at step 1117, the stored question that has the maximum NP relative to the user's query, is identified at 1117A as the stored question which best matches the user's query.

Notably, it can be seen that the second step of the recognition process is much more computationally intensive than the first step above, because several text strings are 50 tokenized, and a comparison is made of several NPs. This would not be practical, nonetheless, if it were not for the fact that the first step has already quickly and efficiently reduced the candidates to be evaluated to a significant degree. Thus, this more computationally intensive aspect of the present 55 invention is extremely valuable, however because it yields extremely high accuracy in the overall query recognition process. In this regard, therefore, this second step of the query recognition helps to ensure the overall accuracy of the system, while the first step helps to maintain a satisfactory speed that provides a real-time feel for the user.

As illustrated in FIG. 11C, the last part of the query/ response process occurs by providing an appropriate matching answer/response to the user. Thus, an identity of a matching stored question is completed at step 1120. Next a 65 file path corresponding to an answer of the identified matching question is extracted at step 1121. Processing continues

26

so that the answer is extracted from the file path at 1122 and finally the answer is compressed and sent to client side system 150 at step 1123.

The discussion above is intended to convey a general overview of the primary components, operations, functions and characteristics of those portions of NLQS system 100 that reside on server side system 180. The discussion that follows describes in more detail the respective sub-systems. Software Modules Used in Server Side System 180

The key software modules used on server-side system 180 of the NLQS system are illustrated in FIG. 5. These include generally the following components: a Communication module 500-identified as CommunicationServer ISAPI 500A (which is executed by SRE Server-side 182—FIG. 1 and is explained in more detail below), and a database process DBProcess module 501 (executed by DB Engine 186—FIG. 1). Natural language engine module 500C (executed by NLE 190-FIG.1) and an interface 500B between the NLE process module 500C and the DBProcess module 500B. As shown here, CommunicationServerISAPI 500A includes a server-side speech recognition engine and appropriate communication interfaces required between client side system 150 and server side system 180. As further illustrated in FIG. 5, server-side logic of Natural Language Query System 100 also can be characterized as including two dynamic link library components: CommunicationServerISAPI 500 and DBProcess 501. The CommunicationServerIASPI 500 is comprised of 3 sub-modules: Server-side Speech Recognition Engine module 500A; Interface module **500**B between Natural Language Engine modules **500**C and DBProcess 501; and the Natural Language Engine modules 500C.

DB Process 501 is a module whose primary function is to connect to a SQL database and to execute an SQL query that is composed in response to the user's query. In addition, this module interfaces with logic that fetches the correct answer from a file path once this answer is passed to it from the Natural Language Engine module 500C.

Speech Recognition Sub-System 182 on Server-Side System

The server side speech recognition engine module 500A is a set of distributed components that perform the necessary functions and operations of speech recognition engine 182 (FIG.1) at server-side 180. These components can be implemented as software routines that are executed by server side 180 in conventional fashion. Referring to FIG. 4A, a more detailed break out of the operation of the speech recognition components 600 at the server-side can be seen as follows:

Within a portion 601 of the server side SRE module 500A, the binary MFCC vector byte stream corresponding to the speech signal's acoustic features extracted at client side system 150 and sent over the communication channel 160 is received. The MFCC acoustic vectors are decoded from the encoded HTTP byte stream as follows: Since the MFCC vectors contain embedded NULL characters, they cannot be transferred in this form to server side system 180 as such using HTTP protocol. Thus the MFCC vectors are first encoded at client-side 150 before transmission in such a way that all the speech data is converted into a stream of bytes without embedded NULL characters in the data. At the very end of the byte stream a single NULL character is introduced to indicate the termination of the stream of bytes to be transferred to the server over the INTERNET 160A using HTTP protocol.

As explained earlier, to conserve latency time between the client and server, a smaller number of bytes (just the 13 MFCC coefficients) are sent from client side system 150 to

27

server side system 180. This is done automatically for each platform to ensure uniformity, or can be tailored by the particular application environment—i.e., such as where it is determined that it will take less time to compute the delta and acceleration coefficients at the server (26 more calculations), than it would take to encode them at the client, transmit them, and then decode them from the HTTP stream. In general, since server side system 180 is usually better equipped to calculate the MFCC delta and acceleration parameters, this is a preferable choice. Furthermore, there is 10 generally more control over server resources compared to the client's resources, which means that future upgrades, optimizations, etc., can be disseminated and shared by all to make overall system performance more reliable and predictable. So, the present invention can accommodate even the worst-case scenario where the client's machine may be quite thin and may just have enough resources to capture the speech input data and do minimal processing.

Dictionary Preparation & Grammar Files Referring to FIG. 4A, within code block 605, various 20 options selected by the user (or gleaned from the user's status within a particular application) are received. For instance, in the case of a preferred remote learning system, Course, Chapter and/or Section data items are communicated. In the case of other applications (such as e-commerce) 25 other data options are communicated, such as the Product Class, Product Category, Product Brand, etc. loaded for viewing within his/her browser. These selected options are based on the context experienced by the user during an interactive process, and thus help to limit and define the 30 scope—i.e. grammars and dictionaries that will be dynamically loaded to speech recognition engine 182 (FIG. 1) for Viterbi decoding during processing of the user speech utterance. For speech recognition to be optimized both grammar and dictionary files are used in a preferred embodiment. A 35 Grammar file supplies the universe of available user queries; i.e., all the possible words that are to be recognized. The Dictionary file provides phonemes (the information of how a word is pronounced—this depends on the specific native language files that are installed—for example, UK English 40 or US English) of each word contained in the grammar file. It is apparent that if all the sentences for a given environment that can be recognized were contained in a single grammar file then recognition accuracy would be deteriorated and the loading time alone for such grammar and dictionary files 45 would impair the speed of the speech recognition process.

To avoid these problems, specific grammars are dynamically loaded or actively configured as the current grammar according to the user's context, i.e., as in the case of a remote learning system, the Course, Chapter and/or Section 50 selected. Thus the grammar and dictionary files are loaded dynamically according to the given Course, Chapter and/or Section as dictated by the user, or as determined automatically by an application program executed by the user.

The second code block **602** implements the initialization 55 of Speech Recognition engine **182** (FIG.1). The MFCC vectors received from client side system **150** along with the grammar filename and the dictionary file names are introduced to this block to initialize the speech decoder.

As illustrated in FIG. 4A, the initialization process 602 ouses the following sub-routines: A routine 602a for loading an SRE library. This then allows the creation of an object identified as External Source with code 602b using the received MFCC vectors. Code 602c allocates memory to hold the recognition objects. Routine 602d then also creates 65 and initializes objects that are required for the recognition such as: Source, Coder, Recognizer and Results Loading of

28

the Dictionary created by code 602e, Hidden Markov Models (HMMs) generated with code 602f; and Loading of the Grammar file generated by routine 602g.

Speech Recognition 603 is the next routine invoked as illustrated in FIG. 4A, and is generally responsible for completing the processing of the user speech signals input on the client side 150, which, as mentioned above, are preferably only partially processed (i.e., only MFCC vectors are computed during the first phase) when they are transmitted across link 160. Using the functions created in External Source by subroutine 602b, this code reads MFCC vectors, one at a time from an External Source 603a, and processes them in block 603b to realize the words in the speech pattern that are symbolized by the MFCC vectors captured at the client. During this second phase, an additional 13 delta coefficients and an additional 13 acceleration coefficients are computed as part of the recognition process to obtain a total of 39 observation vectors O_t referred to earlier. Then, using a set of previously defined Hidden Markov Models (HMMs), the words corresponding to the user's speech utterance are determined in the manner described earlier. This completes the word "recognition" aspect of the query processing, which results are used further below to complete the query processing operations.

It will be appreciated by those skilled in the art that the distributed nature and rapid performance of the word recognition process, by itself, is extremely useful and may be implemented in connection with other environments that do not implicate or require additional query processing operations. For example, some applications may simply use individual recognized words for filling in data items on a computer generated form, and the aforementioned systems and processes can provide a rapid, reliable mechanism for doing so. Once the user's speech is recognized, the flow of SRE 182 passes to Un-initialize SRE routine 604 where the speech engine is un-initialized as illustrated. In this block all the objects created in the initialization block are deleted by routine 604a, and memory allocated in the initialization block during the initialization phase are removed by routine 604b.

Again, it should be emphasized that the above are merely illustrative of embodiments for implementing the particular routines used on a server side speech recognition system of the present invention. Other variations of the same that achieve the desired functionality and objectives of the present invention will be apparent from the present teachings.

Database Processor 186 Operation—DBProcess

Construction of an SQL Query used as part of the user query processing is illustrated in FIG. 4B, a SELECT SQL statement is preferably constructed using a conventional CONTAINS predicate. Module 950 constructs the SQL query based on this SELECT SQL statement, which query is used for retrieving the best suitable question stored in the database corresponding to the user's articulated query, (designated as Question here). A routine 951 then concatenates a table name with the constructed SELECT statement. Next, the number of words present in each Noun Phrase of Question asked by the user is calculated by routine 952. Then memory is allocated by routine 953 as needed to accommodate all the words present in the NP. Next a word List (identifying all the distinct words present in the NP) is obtained by routine 954. After this, this set of distinct words are concatenated by routine 955 to the SQL Query separated with a NEAR () keyword. Next, the AND keyword is concatenated to the SQL Query by routine 956 after each NP. Finally memory resources are freed by code 957 so as to

29

allocate memory to store the words received from NP for any next iteration. Thus, at the end of this process, a completed SQL Query corresponding to the user's articulated question is generated.

Connection to SQL Server—As illustrated in FIG. 4C, 5 after the SQL Query is constructed by routine 710, a routine 711 implements a connection to the query database 717 to continue processing of the user query. The connection sequence and the subsequent retrieved record set is implemented using routines 700 which include the following:

- 1. Server and database names are assigned by routine 711A to a DBProcess member variable
- 2. A connection string is established by routine 711B;
- 3. The SQL Server database is connected under control of 15 code **711**C
- 4. The SQL Query is received by routine 712A
- 5. The SQL Query is executed by code 712B
- 6. Extract the total number of records retrieved by the query-713
- 7. Allocate the memory to store the total number of paired questions-713
- 8. Store the entire number of paired questions into an array—713

Once the Best Answer ID is received at 716 FIG. 4C, from 25 the NLE 14 (FIG. 5), the code corresponding 716C receives it passes it to code in 716B where the path of the Answer file is determined using the record number. Then the file is opened 716C using the path passed to it and the contents of is compressed by code in 716D and prepared for transmission over the communication channel 160B (FIG. 1).

NLQS Database 188—Table Organization

FIG. 6 illustrates a preferred embodiment of a logical structure of tables used in a typical NLQS database 188 35 (FIG.1). When NLQS database 188 is used as part of NLQS query system 100 implemented as a remote learning/training environment, this database will include an organizational multi-level hierarchy that consists typically of a Course 701, which is made of several chapters 702, 703, 704. Each of 40 these chapters can have one or more Sections 705, 706, 707 as shown for Chapter 1. A similar structure can exist for Chapter 2, Chapter 3 . . . Chapter N. Each section has a set of one or more question—answer pairs 708 stored in tables described in more detail below. While this is an appropriate 45 and preferable arrangement for a training/learning application, it is apparent that other implementations would be possible and perhaps more suitable for other applications such as e-commerce, e-support, INTERNET browsing, etc., depending on overall system parameters.

It can be seen that the NLQS database 188 organization is intricately linked to the switched grammar architecture described earlier. In other words, the context (or environment) experienced by the user can be determined at any moment in time based at the selection made at the 55 section level, so that only a limited subset of questionanswer pairs 708 for example are appropriate for section 705. This in turn means that only a particular appropriate grammar for such question-answer pairs may be switched in for handling user queries while the user is experiencing such context. In a similar fashion, an e-commerce application for an INTERNET based business may consist of a hierarchy that includes a first level "home" page 701 identifying user selectable options (product types, services, contact "product types" pages 702, 703, 704, a third page may include particular product models 705, 706, 707, etc., and

30

with appropriate question-answer pairs 708 and grammars customized for handling queries for such product models. Again, the particular implementation will vary from application to application, depending on the needs and desires of such business, and a typical amount of routine optimization will be necessary for each such application. Table Organization

In a preferred embodiment, an independent database is used for each Course. Each database in turn can include 10 three types of tables as follows: a Master Table as illustrated in FIG. 7A, at least one Chapter Table as illustrated in FIG. 7B and at least one Section Table as illustrated in FIG. 7C.

As illustrated in FIG. 7A, a preferred embodiment of a Master Table has six columns—Field Name 70A, Data Type 702A, Size 703A, Null 704A, Primary Key 705A and Indexed 706A. These parameters are well-known in the art of database design and structure. The Master Table has only two fields—Chapter Name 707A and Section Name 708A. Both ChapterName and Section Name are commonly indexed.

A preferred embodiment of a Chapter Table is illustrated in FIG. 7B. As with the Master Table, the Chapter Table has six (6) columns—Field Name 720, Data Type 721, Size 722, Null 723, Primary Key 724 and Indexed 725. There are nine (9) rows of data however, in this case,—Chapter_ID 726, Answer_ID 727, Section Name 728, Answer_Title 729, PairedQuestion 730, AnswerPath 731, Creator 732, Date of Creation 733 and Date of Modification 734.

An explanation of the Chapter Table fields is provided in the file corresponding to the answer is read. Then the answer 30 FIG. 7C. Each of the eight (8) Fields 720 has a description 735 and stores data corresponding to:

> AnswerID 727-an integer that is automatically incremented for each answer given for user convenience

> Section_Name 728—the name of the section to which the particular record belongs. This field along with the AnswerID is used as the primary key

> Answer_Title **729**—A short description of the title of the answer to the user query

> PairedQuestion 730—Contains one or more combinations of questions for the related answers whose path is stored in the next column AnswerPath

> AnswerPath 731—contains the path of a file, which contains the answer to the related questions stored in the previous column; in the case of a pure question/ answer application, this file is a text file, but, as mentioned above, could be a multi-media file of any kind transportable over the data link 160

Creator 732—Name of Content Creator

Date_of_Creation 733—Date on which content was created

Date of Modification 734—Date on which content was changed or modified

A preferred embodiment of a Section Table is illustrated in FIG. 7D. The Section Table has six (6) columns—Field Name **740**, Data Type **741**, Size **742**, Null **743**, Primary Key 744 and Indexed 745. There are seven (7) rows of data-Answer_ID 746, Answer_Tide 747, PairedQuestion 748, AnswerPath 749, Creator 750, Date of Creation 751 and Date of Modification 752. These names correspond to the same fields, columns already described above for the Master Table and Chapter Table.

Again, this is a preferred approach for the specific type of learning/training application described herein. Since the information, etc.), a second level may include one or more 65 number of potential applications for the present invention is quite large, and each application can be customized, it is expected that other applications (including other learning/

31

training applications) will require and/or be better accommodated by another table, column, and field structure/hierarchy.

Search Service and Search Engine—A query text search service is performed by an SQL Search System 1000 shown 5 in FIG. 10. This system provides querying support to process fill-text searches. This is where full-text indexes reside.

In general, SQL Search System determines which entries in a database index meet selection criteria specified by a particular text query that is constructed in accordance with an articulated user speech utterance. The Index Engine 1011B is the entity that populates the Full-Text Index tables with indexes which correspond to the indexable units of text for the stored questions and corresponding answers. It scans through character strings, determines word boundaries, removes all noise words and then populates the full-text 15 index with the remaining words. For each entry in the full text database that meets the selection criteria, a unique key column value and a ranking value are returned as well. Catalog set 1013 is a file-system directory that is accessible only by an Administrator and Search Service 1010. Full-text 20 indexes 1014 are organized into full-text catalogs, which are referenced by easy to handle names. Typically, full-text index data for an entire database is placed into a single

The schema for the full-text database as described (FIG. 7, FIG. 7A, FIG. 7B, FIG. 7C, FIG. 7D) is stored in the tables 1006 shown in FIG. 10. Take for example, the tables required to describe the structure the stored question/answer pairs required for a particular course. For each table Course Table, Chapter Table, Section Table, there are fields—column information that define each parameters that make up the logical structure of the table. This information is stored in User and System tables 1006. The key values corresponding to those tables are stored as Full-Text catalogs 1013. So when processing a full-text search, the search engine returns to the SQL Server the key values of the rows that match the search criteria. The relational engine then uses this information to respond to the query.

As illustrated in FIG. 10, a Full-Text Query Process is implemented as follows:

- A query 1001 that uses a SQL full-text construct 40 generated by DB processor 186 is submitted to SQL Relational Engine 1002.
- 2. Queries containing either a CONTAINS or FREETEXT predicate are rewritten by routine 1003 so that a responsive rowset returned later from Full-Text Provider 1007 will be automatically joined to the table that the predicate is acting upon. This rewrite is a mechanism used to ensure that these predicates are a seamless extension to a traditional SQL Server. After the compiled query is internally rewritten and checked for correctness in item 1003, the query is passed to RUN TIME module 1004. The function of module 1004 is to convert the rewritten SQL construct to a validated run-time process before it is sent to the Full-Text Provider, 1007.
- 3. After this, Full-Text Provider 1007 is invoked, passing 55 the following information for the query:
 - a. A ft_search_condition parameter (this is a logical flag indicating a full text search condition)
 - b. A name of a full-text catalog where a full-text index of a table resides
 - c. A locale ID to be used for language (for example, word breaking)
 - d. Identities of a database, table, and column to be used in the query
 - e. If the query is comprised of more than one full-text 65 construct; when this is the case Full-text provider **1007** is invoked separately for each construct.

32

- 4. SQL Relational Engine 1002 does not examine the contents of ft_search_condition. Instead, this information is passed along to Full-text provider 1007, which verifies the validity of the query and then creates an appropriate internal representation of the full-text search condition.
- The query request/command 1008 is then passed to Querying Support 101A.
- Querying Support 1012 returns a rowset 1009 from Full-Text Catalog 1013 that contains unique key column values for any rows that match the full-text search criteria. A rank value also is returned for each row.
- 7. The rowset of key column values 1009 is passed to SQL Relational Engine 1002. If processing of the query implicates either a CONTAINSTABLE() or FREETEXTTABLE() function, RANK values are returned; otherwise, any rank value is filtered out.
- 8. The rowset values 1009 are plugged into the initial query with values obtained from relational database 1006, and a result set 1015 is then returned for further processing to yield a response to the user.

At this stage of the query recognition process, the speech utterance by the user has already been rapidly converted into a carefully crafted text query, and this text query has been initially processed so that an initial matching set of results can be further evaluated for a final determination of the appropriate matching question/answer pair. The underlying principle that makes this possible is the presence of a full-text unique key column for each table that is registered for full-text searches. Thus when processing a full-text search, SQL Search Service 1010 returns to SQL server 1002 the key values of the rows that match the database. In maintaining these full-text databases 1013 and full text indexes 1014, the present invention has the unique characteristic that the full-text indices 1014 are not updated instantly when the full-text registered columns are updated. This operation is eliminated, again, to reduce recognition latency, increase response speed, etc. Thus, as compared to other database architectures, this updating of the full-text index tables, which would otherwise take a significant time, is instead done asynchronously at a more convenient time. Interface between NLE 190 and DB Processor 188

The result set 1015 of candidate questions corresponding to the user query utterance are presented to NLE 190 for further processing as shown in FIG. 4D to determine a "best" matching question/answer pair. An NLE/ DBProcessor interface module coordinates the handling of user queries, analysis of noun-phrases (NPs) of retrieved questions sets from the SQL query based on the user query, comparing the retrieved question NPs with the user query NP, etc. between NLE 190 and DB Processor 188. So, this part of the server side code contains functions, which interface processes resident in both NLE block 190 and DB Processor block 188. The functions are illustrated in FIG. 4D; As seen here, code routine 880 implements functions to extract the Noun Phrase (NP) list from the user's question. This part of the code interacts with NLE 190 and gets the list of Noun Phrases in a sentence articulated by the user. Similarly, Routine 813 retrieves an NP list from the list of corresponding candidate/paired questions 1015 and stores these questions into an (ranked by NP value) array. Thus, at this point, NP data has been generated for the user query, as well as for the candidate questions 1015. As an example of determining the noun phrases of a sentence such as: "What issues have guided the President in considering the impact off foreign trade policy on American businesses?" NLE 190 would return the following as noun phrases: President,

33

issues, impact of foreign trade policy, American businesses, impact, impact of foreign trade, foreign trade, foreign trade policy, trade, trade policy, policy, businesses. The methodology used by NLE 190 will thus be apparent to those skilled in the art from this set of noun phrases and noun sub-phrases generated in response to the example query.

Next, a function identified as Get Best Answer ID 815 is implemented. This part of the code gets a best answer ID corresponding to the user's query. To do this, routines 813A, 813B first find out the number of Noun phrases for each 10 entry in the retrieved set 1015 that match with the Noun phrases in the user's query. Then routine 815a selects a final result record from the candidate retrieved set 1015 that contains the maximum number of matching Noun phrases.

Conventionally, nouns are commonly thought of as "naming" words, and specifically as the names of "people, places, or things". Nouns such as John, London, and computer certainly fit this description, but the types of words classified by the present invention as nouns is much broader than this. Nouns can also denote abstract and intangible concepts such 20 as birth, happiness, evolution, technology, management, imagination, revenge, politics, hope, cookey, sport, and literacy. Because of the enormous diversity of nouns compared to other parts of speech, the Applicant has found that it is much more relevant to consider the noun phrase as a key 25 linguistic metric. So, the great variety of items classified as nouns by the present invention helps to discriminate and identify individual speech utterances much easier and faster than prior techniques disclosed in the art.

Following this same thought, the present invention also 30 adopts and implements another linguistic entity—the word phrase—to facilitate speech query recognition. The basic structure of a word phrase—whether it be a noun phrase, verb phrase, adjective phrase—is three parts—[pre-Head string],[Head] and [post-Head string]. For example, in the 35 minimal noun phrase—"the children," "children" is classified as the Head of the noun phrase. In summary, because of the diversity and frequency of noun phrases, the choice of noun phrase as the metric by which stored answer is linguistically chosen, has a solid justification in applying this 40 technique to the English natural language as well as other natural languages. So, in sum, the total noun phrases in a speech utterance taken together operate extremely well as unique type of speech query fingerprint.

The ID corresponding to the best answer corresponding to 45 the selected final result record question is then generated by routine 815 which then returns it to DB Process shown in FIG.4C. As seen there, a Best Answer ID I is received by routine 716A, and used by a routine 716B to retrieve an answer file path. Routine 716C then opens and reads the 50 answer file, and communicates the substance of the same to routine 716D. The latter then compresses the answer file data, and sends it over data link 160 to client side system 150 for processing as noted earlier (i.e., to be rendered into audible feedback, visual text/graphics, etc.). Again, in the 55 context of a learning/instructional application, the answer file may consist solely of a single text phrase, but in other applications the substance and format will be tailored to a specific question in an appropriate fashion. For instance, an "answer" may consist of a list of multiple entries corresponding to a list of responsive category items (i.e., a list of books to a particular author) etc. Other variations will be apparent depending on the particular environment.

Natural Language Engine 190 engine 190 is depicted. This engine inplements the word analysis or morphological analysis of words that make up 34

the user's query, as well as phrase analysis of phrases extracted from the query.

As illustrated in FIG. 9, the functions used in a morphological analysis include tokenizers 802A, stemmers 804A and morphological analyzers 806A. The functions that comprise the phrase analysis include tokenizers, taggers and groupers, and their relationship is shown in FIG. 8.

Tokenizer 802A is a software module that functions to break up text of an input sentence 801A into a list of tokens 803A. In performing this function, tokenizer 802A goes through input text 801A and treats it as a series of tokens or useful meaningful units that are typically larger than individual characters, but smaller than phrases and sentences. These tokens 803A can include words, separable parts of word and punctuation. Each token 803A is given an offset and a length. The first phase of tokenization is segmentation, which extracts the individual tokens from the input text and keeps track of the offset where each token originated from in the input text. Next, categories are associated with each token, based on its shape. The process of tokenization is well-known in the art, so it can be performed by any convenient application suitable for the present invention.

Following tokenization, a stemmer process 804A is executed, which can include two separate formsinflectional and derivational, for analyzing the tokens to determine their respective stems 805A. An inflectional stemmer recognizes affixes and returns the word which is the stem. A derivational stemmer on the other hand recognizes derivational affixes and returns the root word or words. While stemmer 804A associates an input word with its stem, it does not have parts of speech information. Analyzer 806B takes a word independent of context, and returns a set of possible parts of speech 806A.

As illustrated in FIG. 8, phrase analysis 800 is the next step that is performed after tokenization. A tokenizer 802 generates tokens from input text 801. Tokens 803 are assigned to parts of a speech tag by a tagger routine 804, and a grouper routine 806 recognizes groups of words as phrases of a certain syntactic type. These syntactic types include for example the noun phrases mentioned earlier, but could include other types if desired such as verb phrases and adjective phrases. Specifically, tagger 804 is a parts-ofspeech disambiguator, which analyzes words in context. It has a built-in morphological analyzer (not shown) that allows it to identify all possible parts of speech for each token. The output of tagger 804 is a string with each token tagged with a parts-of-speech label 805. The final step in the linguistic process 800 is the grouping of words to form phrases 807. This function is performed by the grouper 806, and is very dependent, of course, on the performance and output of tagger component 804.

Accordingly, at the end of linguistic processing 800, a list of noun phrases (NP) 807 is generated in accordance with the user's query utterance. This set of NPs generated by NLE 190 helps significantly to refine the search for the best answer, so that a single-best answer can be later provided for the user's question.

The particular components of NLE 190 are shown in FIG. 4D, and include several components. Each of these components implement the several different functions required in NLE 190 as now explained.

Initialize Grouper Resources Object and the Library 900—this routine initializes the structure variables required to create grouper resource object and library. Specifically, it Again referring to FIG. 4D, the general structure of NL 65 initializes a particular natural language used by NLE 190 to create a Noun Phrase, for example the English natural language is initialized for a system that serves the English

35

language market. In turn, it also creates the objects (routines) required for Tokenizer, Tagger and Grouper (discussed above) with routines 900A, 900B, 900C and 900D respectively, and initializes these objects with appropriate values. It also allocates memory to store all the recognized Noun Phrases for the retrieved question pairs.

Tokenizing of the words from the given text (from the query or the paired questions) is performed with routine 909B—here all the words are tokenized with the help of a local dictionary used by NLE 190 resources. The resultant 10 tokenized words are passed to a Tagger routine 909C. At routine 909C, tagging of all the tokens is done and the output is passed to a Grouper routine 909D.

The Grouping of all tagged token to form NP list is implemented by routine 909D so that the Grouper groups all 15 the tagged token words and outputs the Noun Phrases.

Un-initializing of the grouper resources object and freeing of the resources, is performed by routines 909EA, 909EB and 909EC. These include Token Resources, Tagger Resources and Grouper Resources respectively. After 20 initialization, the resources are freed. The memory that was used to store all Noun Phrases are also de-allocated. Additional Embodiments

In a c-commerce embodiment of the present invention as illustrated in FIG. 13, a web page 1300 contains typical 25 visible links such as Books 1310, Music 1320 so that on clicking the appropriate link the customer is taken to those pages. The web page may be implemented using HTML, a Java applet, or similar coding techniques which interact with the user's browser. For example, if customer wants to buy an 30 album C by Artist Albert, he traverses several web pages as follows: he first clicks on Music (FIG. 13, 1360), which brings up page 1400 where he/she then clicks on Records (FIG. 14, 1450). Alternatively, he/she could select CDs 1460, Videos 1470, or other categories of books 1410, music 35 1420 or help 1430. As illustrated in FIG. 15, this brings up another web page 1500 with links for Records 1550, with sub-categories—Artist 1560, Song 1570, Title 1580, Genre 1590. The customer must then click on Artist 1560 to select the artist of choice. This displays another web page 1600 as 40 illustrated in FIG. 16. On this page the various artists 1650 are listed as illustrated—Albert 1650, Brooks 1660, Charlie 1670, Whyte 1690 are listed under the category Artists 1650. The customer must now click on Albert 1660 to view the albums available for Albert. When this is done, another web page is displayed as shown in FIG. 17. Again this web page 1700 displays a similar look and feel, but with the albums available 1760, 1770, 1780 listed under the heading Tides 1750. The customer can also read additional information 1790 for each album. This album information is similar to 50 the liner notes of a shrink-wrapped album purchased at a retail store. One Album A is identified, the customer must click on the Album A 1760. This typically brings up another text box with the information about its availability, price, shipping and handling charges etc.

When web page 1300 is provided with functionality of a NLQS of the type described above, the web page interacts with the client side and server side speech recognition modules described above. In this case, the user initiates an inquiry by simply clicking on a button designated Contact Me for Help 1480 (this can be a link button on the screen, or a key on the keyboard for example) and is then told by character 1440 about how to elicit the information required. If the user wants Album A by artist Albert, the user could articulate "Is Album A by Brooks available?" in much the 65 same way they would ask the question of a human clerk at a brick and mortar facility. Because of the rapid recognition

36

Filed 09/08/2008

performance of the present invention, the user's query would be answered in real-time by character 1440 speaking out the answer in the user's native language. If desired, a readable word balloon 1490 could also be displayed to see the character's answer and so that save/print options can also be implemented. Similar appropriate question/answer pairs for each page of the website can be constructed in accordance with the present teachings, so that the customer is provided with an environment that emulates a normal conversational human-like question and answer dialog for all aspects of the web site. Character 1440 can be adjusted and tailored according to the particular commercial application, or by the user's own preferences, etc. to have a particular voice style (man, woman, young, old, etc.) to enhance the customer's experience.

In a similar fashion, an articulated user query might be received as part of a conventional search engine query, to locate information of interest on the INTERNET in a similar manner as done with conventional text queries. If a reasonably close question/answer pair is not available at the server side (for instance, if it does not reach a certain confidence level as an appropriate match to the user's question) the user could be presented with the option of increasing the scope so that the query would then be presented simultaneously to one or more different NLEs across a number of servers, to improve the likelihood of finding an appropriate matching question/answer pair. Furthermore, if desired, more than one "match" could be found, in the same fashion that conventional search engines can return a number of potential "hits" corresponding to the user's query. For some such queries, of course, it is likely that real-time performance will not be possible (because of the disseminated and distributed processing) but the advantage presented by extensive supplemental question/answer database systems may be desirable for some users.

It is apparent as well that the NLOS of the present invention is very natural and saves much time for the user and the e-commerce operator as well. In an e-support embodiment, the customer can retrieve information quickly and efficiently, and without need for a live customer agent. For example, at a consumer computer system vendor related support site, a sample diagnostic page might be presented for the user, along with a visible support character to assist him/her. The user could then select items from a "symptoms" page (i.e., a "monitor" problem, a "keyboard" problem, a "printer" problem, etc.) simply by articulating such symptoms in response to prompting from the support character. Thereafter, the system will direct the user on a real-time basis to more specific sub-menus, potential solutions, etc. for the particular recognized complaint. The use of a programmable character thus allows the web site to be scaled to accommodate a large number of hits or customers without any corresponding need to increase the number of human resources and its attendant training issues.

As an additional embodiment, the searching for information on a particular web site may be accelerated with the use of the NLQS of the present invention. Additionally, a significant benefit is that the information is provided in a user-friendly manner through the natural interface of speech. The majority of web sites presently employ lists of frequently asked questions which the user typically wades item by item in order to obtain an answer to a question or issue. For example, as displayed in FIG. 13, the customer clicks on Help 1330 to initiate the interface with a set of lists. Other options include computer related items at 1370 and frequently asked questions (FAQ) at 1380.

As illustrated in FIG. 18, a web site plan for typical web page is displayed. This illustrates the number of pages that

37

have to be traversed in order to reach the list of Frequently-Asked Questions. Once at this page, the user has to scroll and manually identify the question that matches his/her query. This process is typically a laborious task and may or may not yield the information that answers the user's query. The present art for displaying this information is illustrated in FIG. 18. This figure identifies how the information on a typical web site is organized: the Help link (FIG. 13, 1330) typically shown on the home page of the web page is illustrated shown on FIG. 18 as 1800. Again referring to 10 FIG. 18, each sub-category of information is listed on a separate page. For example, 1810 lists sub-topics such as 'First Time Visitors', 'Search Tips', 'Ordering', 'Shipping', 'Your Account' etc. Other pages deal with 'Account information' 1860, 'Rates and Policies' 1850 etc. Down another 15 level, there are pages that deal exclusively with a sub-sub topics on a specific page such as 'First Time Visitors' 1960, 'Frequently Asked Questions' 1950, 'Safe Shopping Guarantee' 1940, etc. So if a customer has a query that is best answered by going to the Frequently Asked Questions link, 20 he or she has to traverse three levels of busy and cluttered screen pages to get to the Frequently Asked Questions page 1950. Typically, there are many lists of questions 1980 that have to be manually scrolled through. While scrolling visually, the customer then has to visually and mentally 25 match his or her question with each listed question. If a possible match is sighted, then that question is clicked and the answer then appears in text form which then is read.

In contrast, the process of obtaining an answer to a question using a web page enabled with the present NLQS 30 can be achieved much less laboriously and efficiently. The user would articulate the word "Help" (FIG. 13, 1330). This would immediately cause a character (FIG. 13, 1340) to appear with the friendly response "May I be of assistance. Please state your question?". Once the customer states the 35 question, the character would then perform an animation or reply "Thank you, I will be back with the answer soon". After a short period time (preferably not exceeding 5-7 seconds) the character would then speak out the answer to the user's question. As illustrated in FIG. 18 the answer 40 would be the answer 1990 returned to the user in the form of speech is the answer that is paired with the question 1950. For example, the answer 1990: "We accept Visa, MasterCard and Discover credit cards", would be the response to the query 2000 "What forms of payments do you accept?"

Another embodiment of the invention is illustrated in FIG. 12. This web page illustrates a typical website that employs NLQS in a web-based learning environment. As illustrated in FIG. 12, the web page in browser 1200, is divided into two or more frames. A character 1210 in the 50 likeness of an instructor is available on the screen and appears when the student initiates the query mode either by speaking the word "Help" into a microphone (FIG. 2B, 215) or by clicking on the link 'Click to Speak' (FIG. 12, 1280). Character 1210 would then prompt the student to select a 55 course 1220 from the drop down list 1230. If the user selects the course 'CPlusPlus', the character would then confirm verbally that the course "CPlusPlus" was selected. The character would then direct the student to make the next selection from the drop-down list 1250 that contains the 60 selections for the chapters 1240 from which questions are available. Again, after the student makes the selection, the character 1210 confirms the selection by speaking. Next character 1210 prompts the student to select 'Section' 1260 of the chapter from which questions are available from the 65 drop down list 1270. Again, after the student makes the selection, character 1210 confirms the selection by articu38

lating the 'Section' 1260 chosen. As a prompt to the student, a list of possible questions appear in the list box 1291. In addition, tips 1290 for using the system are displayed. Once the selections are all made, the student is prompted by the character to ask the question as follows: "Please ask your query now". The student then speaks his query and after a short period of time, the character responds with the answer preceded by the question as follows: "The answer to your question . . . is as follows: . . . ". This procedure allows the student to quickly retrieve answers to questions about any section of the course and replaces the tedium of consulting books, and references or indices. In short, it is can serve a number of uses from being a virtual teacher answering questions on-the-fly or a flash card substitute.

From preliminary data available to the inventors, it is estimate that the system can easily accommodate 100–250 question/answer pairs while still achieving a real-time feel and appearance to the user (i.e., less than 10 seconds of latency, not counting transmission) using the above described structures and methods. It is expected, of course, that these figures will improve as additional processing speed becomes available, and routine optimizations are employed to the various components noted for each particular environment.

Again, the above are merely illustrative of the many possible applications of the present invention, and it is expected that many more web-based enterprises, as well as other consumer applications (such as intelligent, interactive toys) can utilize the present teachings. Although the present invention has been described in terms of a preferred embodiment, it will be apparent to those skilled in the art that many alterations and modifications may be made to such embodiments without departing from the teachings of the present invention. It will also be apparent to those skilled in the art that many aspects of the present discussion have been simplified to give appropriate weight and focus to the more germane aspects of the present invention. The microcode and software routines executed to effectuate the inventive methods may be embodied in various forms, including in a permanent magnetic media, a non-volatile ROM, a CD-ROM, or any other suitable machine-readable format. Accordingly, it is intended that the all such alterations and modifications be included within the scope and spirit of the invention as defined by the following claims.

What is claimed is:

1. A machine executable program for assisting a computing system to effectuate distributed voice query recognition comprising:

- a first audio signal receiving routine for receiving user speech utterance signals representing speech utterances to be recognized, said speech utterances including sentences comprised of one or more words; and
- a first signal processing routine adapted to generate representative speech data values from said speech utterance signals, said representative speech data values being characterized by a first data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech utterance; and
- a formatting routine for rendering said representative speech data values into a transmission format suitable for transmission over a communications channel to a second processing routine executing on a separate computing system
- wherein said representative speech data values are transmitted continuously during said speech utterances within streaming packets and without waiting for silence to be detected and/or said speech utterances to be completed; and

39

- wherein said first data content in said representative speech data values is used by said second processing routine to compute additional data content that when combined with said first data content is sufficient for a speech recognition routine to complete recognition of words articulated in said speech utterance at said separate computing system
- and further wherein an amount of said first data content transmitted is configured and can be varied for said speech utterances based on signal processing capabilities of the computing system and/or transmission characteristics of said communications channel.
- 2. The program of claim 1, wherein said program works within a browser program executing on said computing system as part of a client-server based system.
- 3. The program of claim 1, wherein said first signal ¹⁵ processing routine is further adapted to handle a stream of continuous speech utterances, such that a plurality of representative speech data values are generated for each of said speech utterances in real-time.
- 4. The program of claim 1, wherein said first data content ²⁰ is sufficiently small that said formatting routine can handle in real-time a continuous stream of representative speech data values that may be generated for a corresponding stream of speech utterances.
- 5. The program of claim 1, wherein each of said representative speech data values corresponds to a separate cepstral coefficient value for a corresponding frequency component of said user speech utterance signals, and said first data content corresponds to a set of said frequency components spanning an audible speech frequency range.
- 6. The program of claim 5, wherein said additional data content corresponds to a set of delta and acceleration coefficients computed from a corresponding set of said ceptstral coefficient values.
- 7. The program of claim 6, wherein said additional data content is generated by said second processing routine with less latency than would that resulting if said additional data content were generated by said first signal processing routine.
- 8. The program of claim 1, wherein signal processing functions required to generate said first data content and said additional data content can be allocated as needed between said first signal processing routine operating on a client device and said second signal processing routine operating on a server device based on a determination of computing resources available to said first and second signal processing routines respectively on a client device-by-device basis.
- 9. The program of claim 1, wherein said first signal processing routine is a set of instructions executed by any one of a host microprocessor, an embedded processor, and/or a digital signal processor (DSP).
- 10. The program of claim 1, wherein said first signal processing routine and said first audio signal are implemented as part of a desktop computing system, a portable computing system, a personal digital assistant (PDA), a cell-phone, and/or an electronic interactive toy.
 - 11. A distributed voice recognition system comprising:
 - a sound processing circuit adapted to receive a speech utterance and to generate associated speech utterance signals therefrom; and
 - a first signal processing circuit adapted to generate a first 60 set of speech data values from said speech utterance signals, said first set of speech data values being insufficient by themselves for permitting recognition of words articulated in said speech utterance; and
 - a transmission circuit for formatting and transmitting said 65 first set of speech data values over a communications channel to a second signal processing circuit;

40

- wherein said first set of speech data values are sent in a streaming fashion over said channel before silence is detected and/or said speech utterance is completed; and
- said second signal processing circuit being configured to generate a second set of speech data values based on receiving and processing said speech data values during said speech utterance and before silence is detected, such that second set of speech data values contain sufficient information to be usable by a word recognition engine for recognizing words in said speech utterance:
- and further wherein at least some words are recognized in real-time and output as text before said speech utterance is completed.
- 12. The system of claim 11, wherein said second set of speech data values include said first set of speech data values and a derived set of speech data values, which derived set of speech data values are computed based on said first speech data values.
- 13. The system of claim 12, wherein said first set of data values are MFCC vector coefficients, and said derived set of speech data values are MFCC delta coefficients and a MFCC acceleration coefficients derived from said MFCC vector coefficients.
- 14. The system of claim 11, wherein said second set of speech data values can be generated by said second signal processing circuit in a time that is less than the combination of a first time which would be required by said first signal processing circuit to generate said second set of speech data values from said first set of speech data values combined with a second time which would be required by said transmission circuit to format and transmit said second set of speech data values.
- 15. The system of claim 11, wherein said second set of speech data values can be generated by said second signal processing circuit in less time than that which would be required by said first signal processing circuit to generate said second set of speech data values from said first set of speech data values.
- 16. The system of claim 11, wherein signal processing responsibilities of said first and second signal processing circuits are allocated such that said first signal processing circuit performs less than approximately ½ the required signal processing operations needed to convert said speech utterance signals into a form usable by a word recognition engine.
- 17. The system of claim 11, wherein signal processing functions required to generate said first and second set of speech data values can be allocated between said first signal processing circuit and second signal processing circuit as needed based on computing resources available to said first and second signal processing circuits respectively.
 - 18. The system of claim 11, wherein signal processing functions required to generate said first and second set of speech data values can be allocated between said first signal processing circuit and second signal processing circuit as needed based on computing resources available to said first and second signal processing circuits respectively.
 - 19. The system of claim 11, wherein signal processing functions performed by said first signal processing circuit and second signal processing circuit are configured based on: (i) computing resources available to said first and second signal processing circuits; (ii) performance characteristics of said transmission circuit; and (iii) transmission latencies of said communications channel.
 - 20. The system of claim 11, wherein said first signal processing circuit is also configured to assist said second

41

signal processing circuit with signal processing computations required to generate said second set of speech data values.

- 21. The system of claim 11, wherein said first set of speech data values represent the least amount of data that can used by said second signal processing circuit to generate said second set of data values usable for a word recognition process.
- 22. A system for recognizing speech information, comprising:
 - storage means for storing one or more words to be recognized by the system based on the speech information; and
 - means for capturing speech signals corresponding to the speech information; and
 - first processing means for generating partially recognized speech data from said speech signals, said first processing means performing a first signal processing operation on said speech signals, said first signal processing operation being insufficient to permit said partially recognized speech data to be correlated with said one or more words; and
 - second processing means for generating fully recognized speech data from said partially recognized speech data, using a second signal processing operation, such that said fully recognized speech data can be correlated with said one or more words, said second processing means being distinct and physically separated from said first processing means; and
 - third processing means for generating recognized sentence data from said one or more words using natural language processing operations including word phrase analysis performed on said one or more words;
 - a non-permanent data transmission connection coupling said first and second processing means;
 - transmitting means for transmitting said partially processed speech data signals from said first processing means through said non-permanent data transmission connection to said second processing means, said transmitting means using a continuously generated byte 40 stream that is transmitted while a speech utterance is occurring and before silence is detected;
 - wherein the system recognizes a complete sentence included in the speech information based on said recognized sentence data and determines a best response 45 to said complete sentence in real-time.
- 23. The system of claim 22, wherein said first processing means is located at a client site, and said second processing means is located at a remote server site.
- **24**. The system of claim **23**, wherein said storage means 50 is also located at said server site.
- 25. The system of claim 22, wherein said non-permanent connection is either a circuit switched or packet switched connection.
- 26. The system of claim 22, wherein said non-permanent 55 connection includes an intranet network linking said first and second signal processing means.
- 27. The system of claim 22, wherein said non-permanent connection is a wireless communications channel.
- 28. The system of claim 22, wherein said first signal 60 processing operation includes an operation for extracting spectral parameter vectors from said speech signals.
- 29. The system of claim 28, wherein said first signal processing operation further includes an operation for decomposing the spectral parameter vectors with a Mel-65 frequency transfer process to obtain MFCC coefficients for said spectral parameter vectors.

42

- **30**. The system of claim **28**, wherein said partially recognized speech data comprises an observation vector O_r which includes said MFCC coefficients and delta and acceleration coefficients obtained from said spectral parameter vectors.
- 31. The system of claim 1, wherein said partially recognized speech data includes an observation vector O_r, and second signal processing operation includes a Viterbi decoding operation for mapping a sequence of said observation vectors O_r, with a speech data symbol.
- 32. The system of claim 10, wherein said second signal processing operation further includes a text conversion operation for converting said speech data symbol into one or more text words.
- 33. The system of claim 22, wherein said second processing means further utilizes environment variables for determining which of said one or more text words correspond to such speech information.
- **34**. A method of performing speech recognition comprising the steps of:
 - (a) receiving user speech utterance signals representing speech utterances to be recognized, said speech utterances including sentences comprised of one or more words; and
- (b) generating representative speech data values with a first computing device which by themselves are insufficient for completing recognition of said one or more words contained in said speech utterance signals; and
- (c) formatting said representative speech data values into a transmission format suitable for transmission over a communications channel from said first computing device to a second computing device; and
 - wherein said representative speech data values are transmitted continuously during said speech utterances within streaming packets and without waiting for silence to be detected and/or said speech utterances to be completed; and
- (d) performing a recognition of said one or more words at said second computing device using said representative speech data values and additional speech data values derived from said representative speech data values to generate recognized text;
- (e) performing a natural language processing operation on said recognized text to determine a meaning associated with said sentences in real-time.
- 35. The method of claim 34, wherein recognition of said one or more words is achieved with less latency than that resulting if said recognition were performed entirely by said first computing device.
- 36. The method of claim 34, wherein signal processing functions required to perform said recognition can be allocated and configured between said first computing device and said second computing device as needed based on an evaluation of computing resources available to said first and second computing devices respectively.
- 37. The method of claim 34, wherein said first computing device is part of a client computing system, and said second computing device is part of a server computing system, and said communications channel is a network.
- **38**. A speech recognition program operating on a server coupled through a network to a client device, the program comprising:
 - a receiving routine for receiving speech data from the client device, said speech data being associated with a speech utterance from a user of the client device;
 - wherein said speech data has a data content that is adjusted based on signal processing capabilities of the

n r s i

43

- client device and resources available to the server for processing speech data;
- further wherein said speech data is transmitted continuously by the client device during said speech utterance within streaming packets and without waiting for 5 silence to be detected and/or said speech utterance to be completed;
- a speech recognition processing routine, for recognizing words contained in said speech utterance;
- wherein when said speech data from the client device is insufficient for recognizing words, additional speech related data is computed by the server to generate additional speech data to be combined with said speech data as an input to said speech recognition processing 15 routine;
- a natural language processing routine, for recognizing word sentences contained in said speech utterance by performing one or more natural language processing operations on words contained in said speech utterance. 20
- 39. The program of claim 38, wherein said data content is also adjusted by considering transmission characteristics of the network, and a transmission speed capability of the client
- 40. The program of claim 38, wherein a calibration 25 routine automatically determines an optimal setting for said data content.
- 41. The program of claim 40, wherein said calibration routine executes on the client device.
- includes mel frequency cepstral coefficients (MFCCs), and said additional speech related data includes MFCC delta coefficients and a MFCC acceleration coefficients derived from said MFCC vector coefficients.
- 43. The program of claim 38, wherein the network 35 includes one of an INTERNET connection, a LAN connection, a wireless connection, or a combination thereof.
- 44. The program of claim 38, wherein said words contained in said speech utterance are recognized in real-time by said speech recognition routine.
- 45. The program of claim 44, wherein said word sentences contained in said speech utterance are recognized in realtime by said natural language processing routine.
- 46. The program of claim 38, wherein the client device personal digital assistant, and a cell phone.
- 47. The program of claim 38, wherein said natural language processing operations include word phrase analysis, including a noun phrase analysis.
- **48**. A method of performing speech recognition at a server 50 coupled to a client device through a network, the method comprising the steps of:
 - (a) determining signal processing capabilities of the client device available for processing speech data associated with a user speech utterance;
 - (b) determining resources available to the server for processing speech related data that is to be transmitted by the client device to the server;
 - (c) configuring a data content to be used for transmitting said speech related data through the network based on the results of steps (a) and (b);

44

- (d) receiving said speech related data with said data content at the server continuously from the client device during said speech utterance within streaming packets and without waiting for silence to be detected and/or said speech utterance to be completed;
- (e) processing said speech related data at the server so that additional speech related data is computed at the server from said speech data and is used to augment said speech related data when said speech related data contains acoustic feature data from said speech utterance that is insufficient for the server to perform accurate recognition of words contained in said speech
- (f) generating a speech data observation vector suitable for processing by a speech recognition routine, said speech data observation vector being based on said speech related data as received from the client device, and/or said speech related data as augmented by the server:
- (g) recognizing words contained in said speech utterance with a speech recognition engine using said speech data observation vector;
- (h) recognizing word sentences contained in said speech utterance by performing one or more natural language processing operations on words contained in said speech utterance.
- 49. The method of claim 48, wherein said data content is 42. The program of claim 38, wherein said speech data 30 also configured by considering transmission characteristics of the network, and a transmission speed capability of the client device.
 - 50. The method of claim 48, further including a step of executing a calibration routine to automatically determine an optimal setting for said data content.
 - 51. The method of claim 50, wherein said calibration routine executes on the client device.
 - 52. The method of claim 48, wherein said acoustic feature data includes mel frequency cepstral coefficients (MFCCs), and said additional speech related data includes MFCC delta coefficients and a MFCC acceleration coefficients derived from said MFCC vector coefficients.
- 53. The method of claim 48, wherein the network includes includes one of a desktop computer, a portable computer, a 45 one of an INTERNET connection, a LAN connection, a wireless connection, or a combination thereof.
 - 54. The method of claim 48, wherein said words contained in said speech utterance are recognized in real-time by said speech recognition routine.
 - 55. The method of claim 54, wherein said word sentences contained in said speech utterance are recognized in realtime by said natural language processing routine, and a response is sent to the client device in real-time.
 - 56. The method of claim 48, wherein the client device includes one of a desktop computer, a portable computer, a personal digital assistant, and a cell phone.
 - 57. The method of claim 48, wherein said natural language processing operations include word phrase analysis, including a noun phrase analysis.

KENT DECLARATION EXHIBIT 2



(12) United States Patent Bennett et al.

US 6,665,640 B1 (10) Patent No.: (45) Date of Patent: Dec. 16, 2003

(54) INTERACTIVE SPEECH BASED LEARNING/ TRAINING SYSTEM FORMULATING SEARCH QUERIES BASED ON NATURAL LANGUAGE PARSING OF RECOGNIZED **USER QUERIES**

(75) Inventors: Ian M. Bennett, Palo Alto, CA (US); Bandi Ramesh Babu, Anantapur Gulbarga (IN); Kishor Morkhandikar,

Bangalore (IN); Pallaki Gururaj,

Bangalore (IN)

Assignee: Phoenix Solutions, Inc., Palo Alto, CA

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 09/439,173

Filed: Nov. 12, 1999

Int. Cl.⁷ G10L 15/18; G10L 15/22; G06F 17/30 **U.S. Cl.** **704/257**; 704/270.1; 704/275;

707/4 (58)Field of Search 704/257, 270.1, 704/275; 707/4

(56)References Cited

U.S. PATENT DOCUMENTS

4,473,904 A	9/1984	Suehiro et al 381/36
4,587,670 A	5/1986	Levinson et al 381/43
4,783,803 A	11/1988	Baker et al 381/42
4,785,408 A	11/1988	Britton et al.

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP 1 094 388 A2 4/2001 1 096 471 A1 EP 5/2001 WO 9811534 3/1998

(List continued on next page.)

OTHER PUBLICATIONS

L. Travis, "Handbook of Speech Pathology," Appleton-Century-Crofts, Inc., 1957, pp. 91-124.

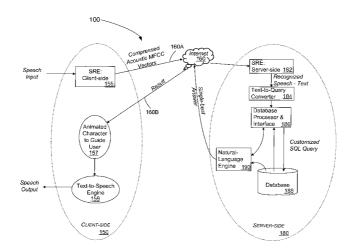
(List continued on next page.)

Primary Examiner—Tālivaldis Ivars Ŝmits (74) Attorney, Agent, or Firm-J. Nicholas Gross

ABSTRACT (57)

A real-time speech-based learning/training system distributed between client and server, and incorporating speech recognition and linguistic processing for recognizing a spoken question and to provide an answer to the student in a learning or training environment implemented on an intranet or over the Internet, is disclosed. The system accepts the student's question in the form of speech at his or her computer, PDA or workstation where minimal processing extracts a sufficient number of acoustic speech vectors representing the utterance. The system as implemented accepts environmental variables such as course, chapter, section as selected by the user so that the search time, accuracy and response time for the question can be optimized. A minimum set of acoustic vectors extracted at the client are then sent via a communications channel to the server where additional acoustic vectors are derived. Using Hidden Markov Models (HMMs), and appropriate grammars and dictionaries conditioned by the course, chapter and section selections made by the student, the speech representing the user's query is fully decoding to text at the server. This text corresponding to the user's query is then simultaneously sent to a natural language engine and a database processor where an optimized SQL statement is constructed for a full-text search from a SQL database for a recordset of several stored questions that best matches the user's query. Further processing in the natural language engine narrows the search down to a single stored question. The answer that is paired to this single stored question is then retrieved from the file path and sent to the student computer in compressed form. At the student's computer, the answer is articulated using a text-to-speech engine in his or her native natural language. The system requires no training and can operate in several natural languages.

38 Claims, 31 Drawing Sheets



US 6,665,640 B1

Page 2

	U.S. PATENT	DOCUMENTS	WO WO 01/18693 A2 3/2001
4,852,170	η Δ 7/1989	Bordeaux	WO WO 01/26093 A1 4/2001
4,914,590		Loatman et al 364/419	WO WO 01/78065 A1 10/2001 WO WO 01/95312 A1 12/2001
4,991,094		Fagan et al	
4,991,21		Garrett et al 381/43	WO WO 02/03380 A1 1/2002
5,068,789		van Vliembergen 364/419	OTHER PUBLICATIONS
5,146,40		Church	OTTEN TOBEROTHION
5,157,72		Schloss	L.E. Baum, T. Petrie, "Statistical Inference for Probabilistic
5,231,670		Goldhor et al 381/43	Functions for Finite State Markov Chains," The Annals of
5,293,584		Brown et al 395/2.86	Mathematical Statistics, 37: 1554–1563, 1966.
5,384,892		Strong	P. Lieberman, "Intonation, Perception and Language,"
5,475,792		Stanford et al 395/2.42	Research Monograph No. 38, MIT Press, Cambridge, Mass.,
5,513,298		Stanford et al 395/2.52	1967, pp. 5–37.
5,540,589		Waters 434/156	L.E. Baum et al, "A Maximization Technique Occurring in
5,602,963		Bissonnette et al 395/2.84	the Statistical Analysis of Probabilistic Functions of Markov
5,615,290		Stanford et al 704/270.1	Chains," The Annals of Mathematical Statistics, 1970, vol.
5,680,628		Carus et al 395/759	41, No. 1, pp. 164–171.
5,727,950		Cook et al 434/350	J.L. Flanagan, "Speech Analysis Synthesis and Perception,"
5,758,322	2 A 5/1998	Rongley 704/275	2 nd edition, Springer-Verlag Berlin, 1972, pp. 1–53.
5,802,250	6 A 9/1998	Fawcett et al 707/104	
5,819,220	O A 10/1998	Sarukkai et al 704/243	L.E. Baum, "An Inequality and Associated Maximation
5,867,81	7 A 2/1999	Catallo et al 704/255	Technique in Statistical Estimation for Probabilistic Func-
5,873,062	2 A 2/1999	Hansen et al.	tions of Markov Processes," <i>Inequalities–III</i> , pp. 1–8, 1972.
5,915,230	6 A 6/1999	Gould et al 704/251	G.D. Forney, "The Viterbi Algorithm," Proceedings of the
5,956,683	3 A 9/1999	Jacobs et al 704/275	<i>IEEE</i> , vol. 73, pp. 268–278, Mar. 1973.
5,960,394		Gould et al 704/240	J.H. Baker, "The Dragon System—An Overview," IEEE
5,960,399		Barclay et al 704/270	Trans. on ASSP Processing, ASSP-23(1): 24-29, Feb. 1975.
5,983,190		Trower, II et al 704/276	I. Bennett, "A Study of Speech Compression Using Analog
5,995,928		Nguyen et al.	Time Domain Sampling techniques," A Dissertation Sub-
6,009,38		Ramaswamy et al.	mitted to the Dept. Of Electrical Engineering and the
6,029,12		Gillick et al 704/200	Committee on Graduate Studies of Stanford University, May
6,035,275		Brode et al.	1975, pp. 16–32; 76–111.
6,119,08		Kuhn et al.	H.R. Rabiner, "Digital Processing of Speech Signals," Pren-
6,138,089		Guberman 704/207	tice Hall, 198, pp. 116–171; 355–395.
6,141,640			F. Jelinek et al, "Continuous Speech Recognition Statistical
6,144,848		Walsh et al 455/419	Methods," Handbook of Statistics, vol. 2, P.R. Krishnaiah,
6,144,938		Surace et al	Ed. Amsterdam. The Netherlands, North–Holland, 1982, pp.
6,256,60		Digalakis et al.	549–573.
6,269,336		Ladd et al.	L.R. Bahl, F. Jeninek, R.L. Mercer, "A Maximum Likeli-
6,327,568 6,356,869			hood Approach to Continuous Speech Recognition," <i>IEEE</i>
6,363,349		Chapados et al 704/275 Urs et al.	Transactions on Pattern Analysis and Machine Intelligence,
6,374,219			PAMI-5, pp. 179–190, Mar. 1983.
6,381,59		Eichstaedt et al.	R.A. Hudson, "Word Grammar," Blackman Inc., Cam-
6,389,389		Meunier et al.	
6,408,272		White et al.	bridge, MA, 1984, pp. 1–14; 41–42; 76–90; 94–98;
6,411,920		Chang	106–109; 211–220.
6,427,063		Cook et al.	R. Quirk, S. Greenbaum, G. Leech and J. Svartvik, "A
2001/0032083		Van Cleven	Comprehensive Grammar of English Language," Longman,
2001/0056346	5 A1 12/2001	Ueyama et al.	London and New York, 1985, pp. 245–331.
2002/0046023		Fujji et al.	J. Makhoul, S. Roucos, H. Gish, "Vector Quantization in
2002/0059068		Rose et al.	Speech Coding," <i>Proceedings of the IEEE</i> , vol. 73, No. 11,
2002/0059069	9 A1 5/2002	Hsu et al.	Nov. 1985, pp. 1551–1588.
2002/0086269	9 A1 7/2002	Shpiro	L. Rabiner, "A Tutorial on Hidden Markov Models and
2002/0087325	5 A1 7/2002	Lee et al.	Selected Applications in Speech Recognition," <i>Proceedings</i>
2002/008765	5 A1 7/2002	Bridgman et al.	of the IEEE, vol. 77, No. 2, Feb. 1989, pp. 257–286.
2002/009152	7 A1 7/2002	Shiau	A. Gersho and R.M. Gray, "Vector Quantization and Com-
			pression," Kluwer Academic Publishers, 1991, pp. 309–340.
FOREIGN PATENT DOCUMENTS		NT DOCUMENTS	H.R. Rabiner and B.H. Juang, "Fundamentals of Speech
WO	0.00/40044 44	0/4000	Recognition," Prentice Hall, 1993, pp. 11-68.
	O 99/48011 A1	9/1999	Nelson Morgan, Herv' Bourlard, Steve Renals, Michael
	O 00/14727 A1	3/2000 3/2000	Cohen and Horacio Franco, "Hybrid Neural Network/Hid-
	O 00/17854 A1	3/2000 4/2000	den Markov Model Systems for Continuous Speech Recog-
	O 00/20962 A2	4/2000 4/2000	nition," Journal of Pattern Recognition and Artificial Intel-
	O 00/21075 A1 O 00/21232 A2	4/2000 4/2000	ligence, vol. 7, No. 4, 1993, pp. 899–916.
	O 00/21232 A2 O 00/22610 A1	4/2000	L. Travis, "Handbook of Speech Pathology", Appleton-Cen-
	O 00/32010 A1	5/2000	tury–Crofts, Inc., 1957.
	O 00/30287 A1	5/2000	L.E. Baum, T. Petrie, "Statistical inference for probablistic
	O 00/68823 A2	11/2000	functions for finite state Markov chains", Ann. Math. Stat.,
	O 01/16936 A1	3/2001	37: 1554–1563, 1966.
	,	•	

US 6,665,640 B1

Page 3

- J.L. Flanagan, "Speech Analysis Synthesis and Perception", 2nd edition, Springer-Verlag Berlin, 1972
- L.E. Baum, "An inequality and associated maximation technique in statistical estimation for probablistic functions of Markov processes", Inequalities 3: 1-8, 1972.
- R. Quirk, S. Greenbaum, G. Leech and J. Svartik, "A Comprehensive Grammar of English Language," Longman, London and New York, 1985, pp. 245–331. J. Makhoul, S. Roucos, H. Gish, "Vector Quantitization in
- Speech Coding," Proceedings of the IEEE, vol. 73, No. 11, Nov. 1985, pp. 1551–1588.
- L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of the IEEE, vol. 77, No. 2, Feb. 1989, pp. 257–286. A. Gersho and R.M. Gray, "Vector Quantization and Com-
- pression," Kluwer Academic Publishers, 1991, pp. 309-340. H.R. Rabiner and B.H. Juang, "Fundamentals of Speech Recognition," Prentice Hall, 1993, pp. 11–68.
- Nelson Morgan, Herv" Bourlard, Steve Renals, Michael Cohen and Horacio Franco, "Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition," Journal of Pattern Recognition and Artificial Intelligence, vol. 7, No. 4, 1993, pp. 899-916.
- G.D. Forney, "The Viterbi Algorithm," Proceedings of the
- IEEE, vol. 73, pp. 268–278, Mar. 1973. J.H. Baker, "The Dragon System—An Overview," IEEE Trans. on ASSP Processing, ASSP-23(1): Feb. 24-29, 1975. I. Bennett, "A Study of Speech Compression Using Analog Time Domain Sampling techniques," A Dissertation Sub-mitted to the Dept. Of Electrical Engineering and the Committee on Graduate Studies of Stanford University, May 1975, pp. 16–32; 76–111.
- H.R. Rabiner, "Digital Processing of Speech Signals," Pren-
- tice Hall, 1978, pp. 116–171; 355–395. F. Jelinek et al, "Continuous Speech Recognition: Statistical Methods," Handbook of Statistics, vol. 2, P.R. Krishnaiah, Ed. Amsterdam, The Netherlands, North-Holland, 1982, pp. 549-573.
- L.R. Bahl, F. Jeninek, R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-5, pp. 179-190, Mar. 1983.
- R.A. Hudson, "Word Grammar," Blackman Inc., Cambridge, MA, 1984, pp. 1-14; 41-42; 76-90; 94-98; 106-109; 211-220.
- J.D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, NJ. 1980.
- I. Bennett and J. Linvill, "A Study of Time Domain Speech Compression by Means of a new Analog Speech Processor", Journal of the Audio Engineering Society, vol. 23, No. 9, 1975.
- R. Quirk, S. Greenbaum, G. Leech and J. Svartvik, A Comprehensive Grammar of English Language, Longman, London and New York, 1985.
- I. Guyon and P. Wang editors Advances in Pattern Recognition Systems using Neural Networks, vol. 7 of a Series in Machine Perception and Artificial Intelligence, World Scientific, Feb. 1994.
- P. Lieberman, "Intonation, Perception and Language", Research Monograph No. 38, MIT Press, Cambridge, Mass. Arons, B., "The Design of Audio Servers and Toolkits for Supporting Speech in the User Interface," believed to be Published in: Journal of the American Voice I/O Society, pp. 27-41, Mar. 1991.

- Hazen,T et al., "Recent Improvements in an Approach to Segment-Based Automatic Language Identification,' believed to be published in: Proceedings of the 1994 International Conference on Spoken Language Processing, Yokohama, Japan, pp. 1883-1886, Sep. 1994.
- House, D., "Spoken-Language Access to Multimedial (SLAM): A Multimodal Interface to the World-Wide-Web, Masters Thesis, Oregon Graduate Institute, Department of Computer Science & Engineering, 59 pages, Apr. 1995.
- Julia, L. et al., "Http://www.speech SRI.com/Demos/ Atis.hbelieved to be published in: Proceedings AAAI'97: Stanford, pp. 72-76, Jul. 1997.
- Lau, R. et al, "Webgalaxy-Integrating Spoken Language and Hypertext Navigation," believed to be published in: in Kokkinakis, G. et al., (Eds.) Eurospeech '97, Proceedings of the 5th European Conference on Speech Communication and Technology, Rhodes (Greece), Sep. 22-25, 1997, pp. 883-886, 1997.
- Digalakis, V. et al., "Product-Code Vector Quantization of Cepstral Parameters for Speech Recognition over the WWW," believed to be published in: Proc. ICSLP '98, 4 pages, 1998.
- Melin, H., "On Word Boundary Detection in Digit-Based Speaker Verification," believed to be published in: Workshop on Speaker Recognition and its Commercial and Forensic Applications (RLA2C), Avignon, France, Apr. 20-23, pp. 46-49, 1998.
- Ramaswamy, G. et al., "Compression of Acoustic Features for Speech Recognition in Network Environments," believed to be published in: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 977-980, Jun. 1998.
- Lu, B. et al., "Scalability Issues in the Real Time Protocol (RTP)," Project Report for CPSC 663 (Real Time Systems), Dept. of Computer Science, Texas A & M University, 19
- Giuliani, D. et al., "Training of HMM with Filtered Speech Material for Hands-Free Recognition," believed to be published in: Proceedings of ICASSP '99, Phoenix, USA, 4 pages, 1999.
- Digalakis, V. et al., "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web," believed to be published in: IEEE Journal on Selected Areas of Communications, 22 pages, 1999.
- Tsakalidis, S. et al., "Efficient Speech Recognition Using Subvector Quantization and Discrete-Mixture HMMs, believed to be published in: Proc. ICASSP '99, 4 pages,
- Lin, B. et al., "A Distributed Architecture for Cooperative Spoken Dialogue Agents with Coherent Dialogue State and History," believed to be published in: IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado, USA, 4 pages, Dec. 1999.
- Meunier, J., "RTP Payload Format for Distributed Speech Recognition," 48th IETF AVT WG—Aug. 3, 2000, 10 pages, 2000.
- Sand Cherry Networks, SoftServer product literature, 2 pages, 2001.
- Kim, H. et al., "A Bitstream-Based Front-End for Wireless Speech Recognition on IS-136 Communications System, IEEE Transactions on Speech and Audio Processing, vol. 9, No. 5, pp. 558-568, Jul. 2001.
- * cited by examiner

U.S. Patent

Dec. 16, 2003

Sheet 1 of 31

US 6,665,640 B1

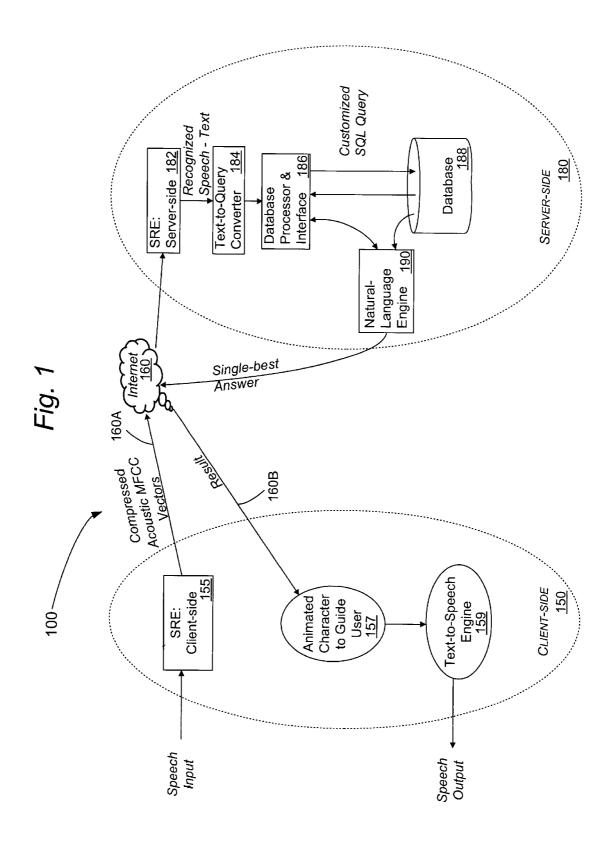


Figure 2 (Page 1/3)

CLIENT-SIDE SYSTEM LOGIC

U.S. Patent

Sheet 2 of 31

US 6,665,640 B1

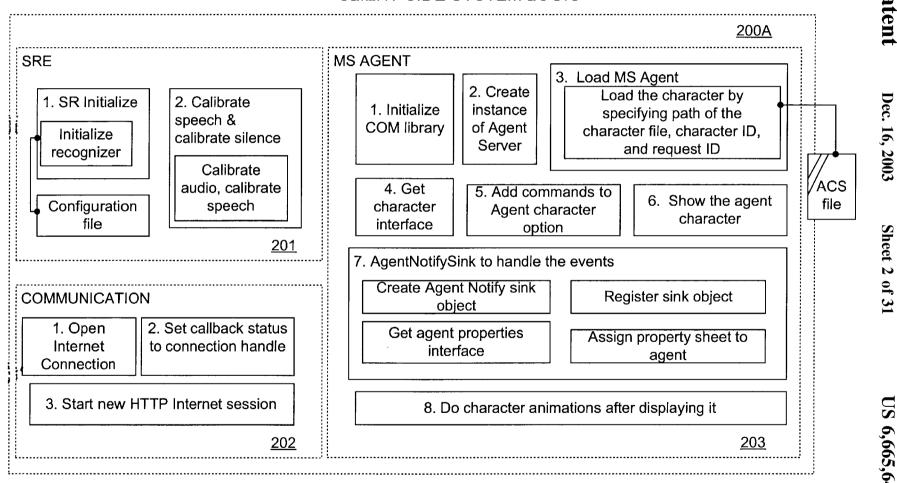


Figure 2 (Page 2/3)

CLIENT-SIDE SYSTEM LOGIC

U.S. Patent

Dec. 16, 2003

Sheet 3 of 31

US 6,665,640 B1

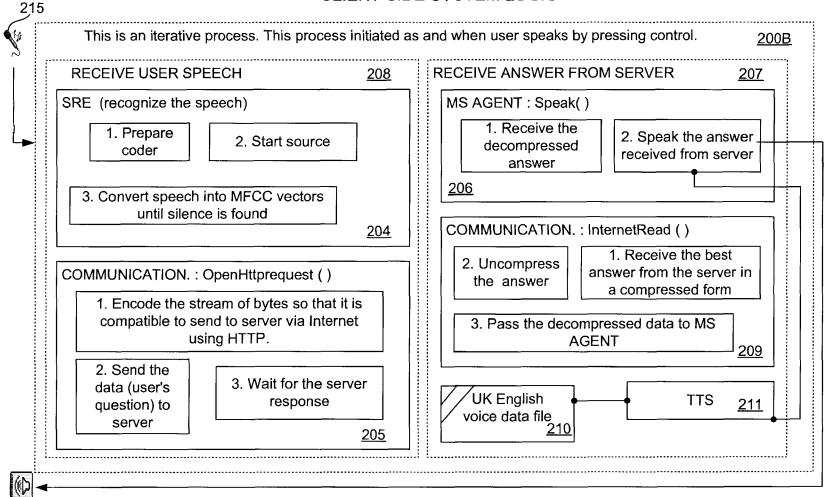


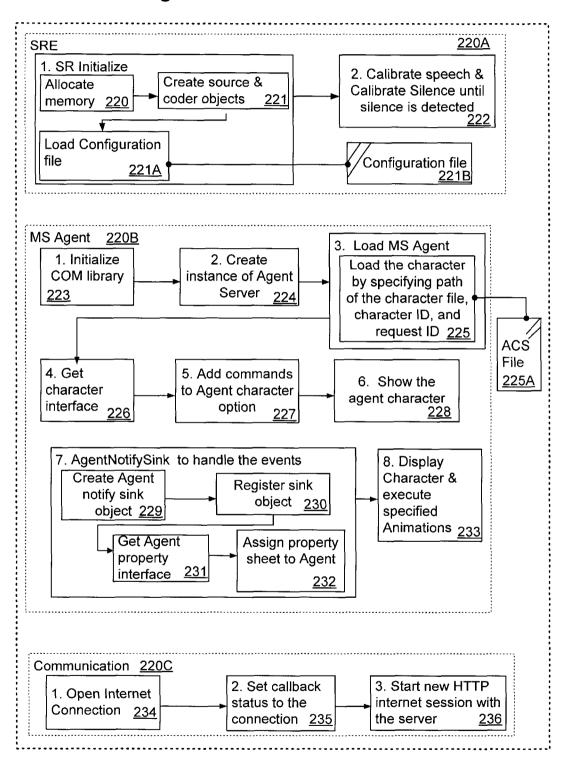
Figure 2 (Page 3/3)

CLIENT-SIDE SYSTEM LOGIC

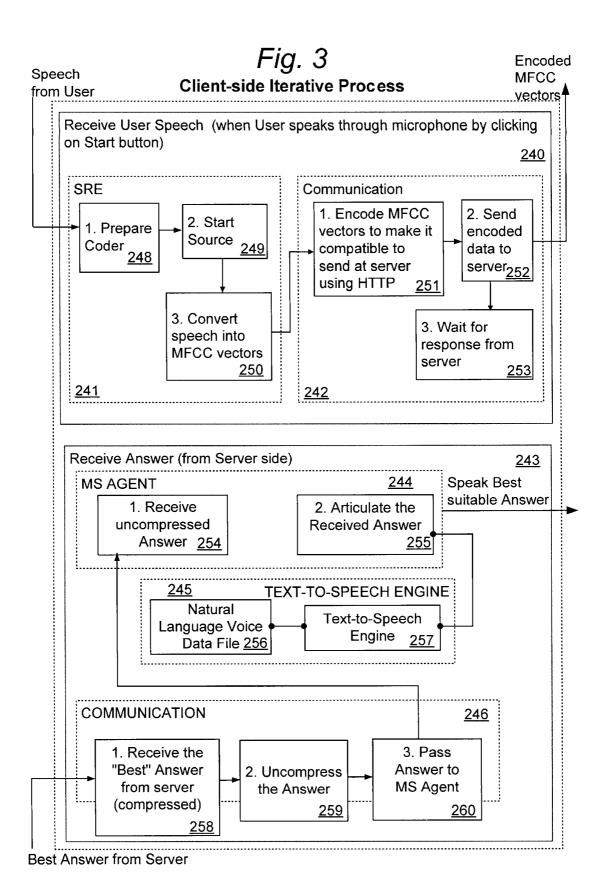
UN-INITIALIZATION (performs as and when user quits i.e. closes the web page) 200C COMMUNICATION 213 MS AGENT **SRE** 212 1. Release 2. Release 3. Unload 1. Close the Internet commands character agent handle i.e. the Interface Interface a. Delete the objects connection created while established with initialization process server. 4. Release 5. Release 6. Unregister b. Deallocate the AgentNotifysink prop. sheet Agent Notifysink memory assigned to Interface Interface the structure, which will be holding the 2. Close the Internet parameters for session which is speech created at the time of initialization 7. Release Agent Interface 214

U.S. Patent Dec. 16, 2003 Sheet 5 of 31 US 6,665,640 B1

Fig. 2-2 Client-side Initialization



U.S. Patent Dec. 16, 2003 Sheet 6 of 31 US 6,665,640 B1



U.S. Patent Dec. 16, 2003 Sheet 7 of 31 US 6,665,640 B1

Fig. 4
Client-side Un-Initialization

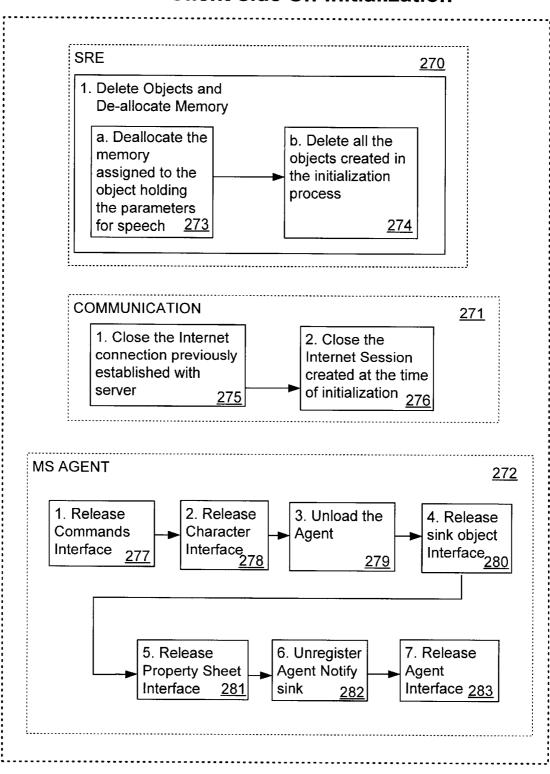


Fig. 4A

Document 52-3

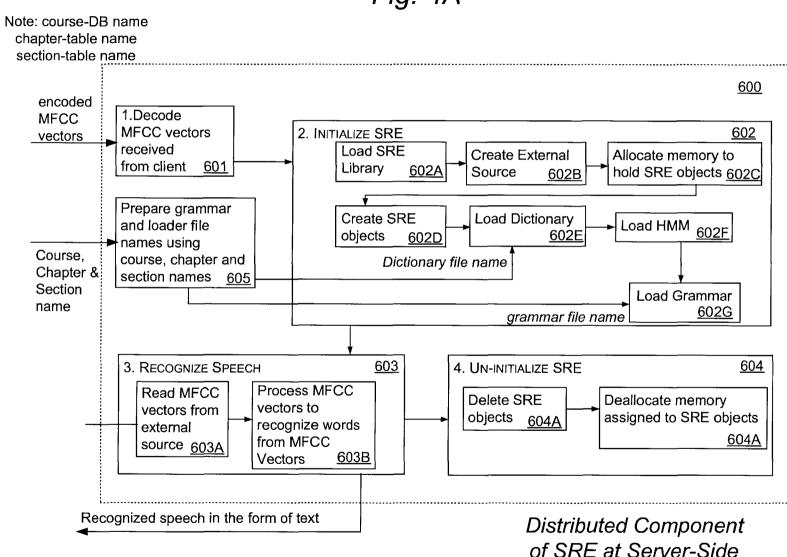
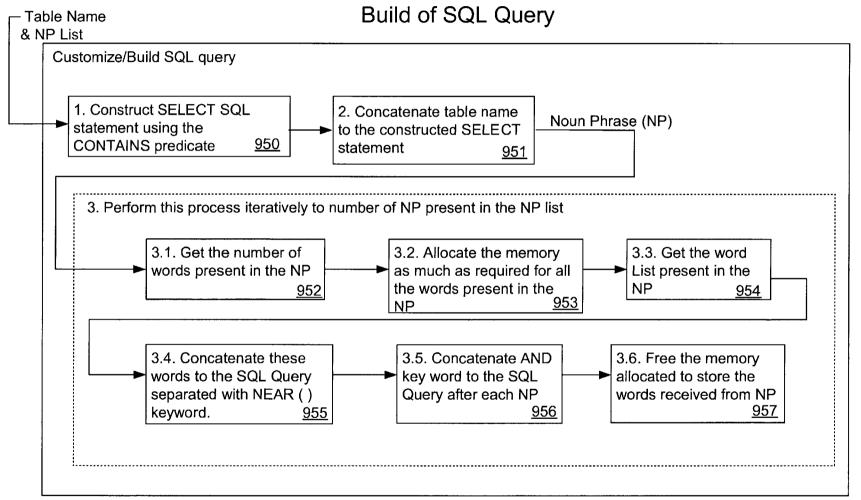


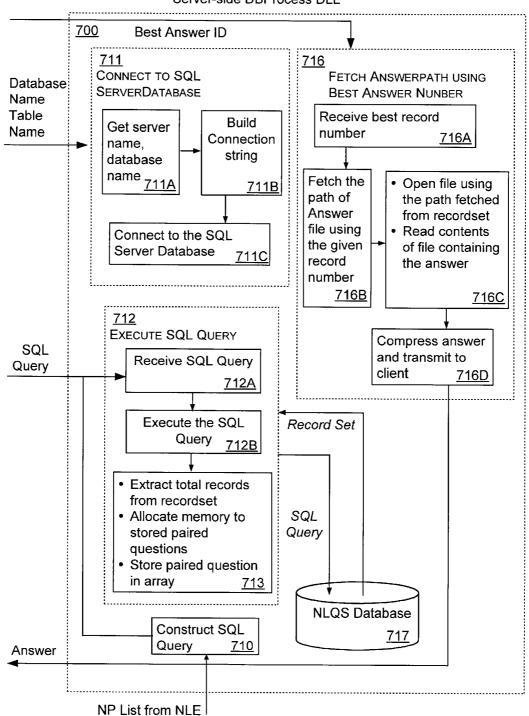
Fig. 4B

Build of SQL Query

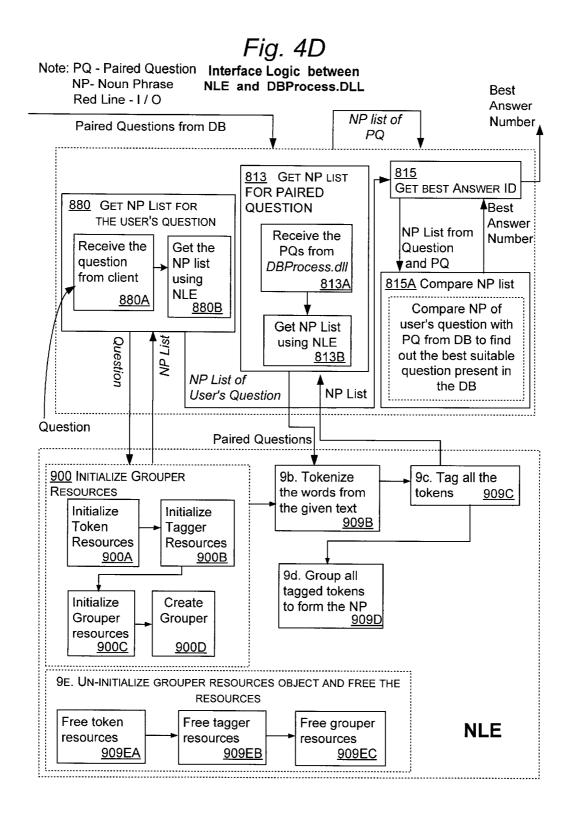


U.S. Patent Dec. 16, 2003 Sheet 10 of 31 US 6,665,640 B1

Fig. 4C
Server-side DBProcess DLL

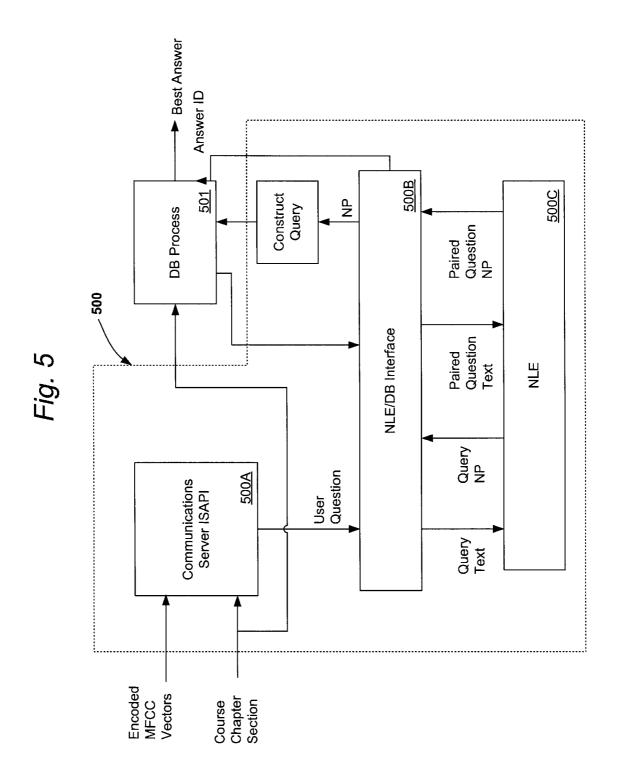


U.S. Patent Dec. 16, 2003 Sheet 11 of 31 US 6,665,640 B1



Dec. 16, 2003

Sheet 12 of 31

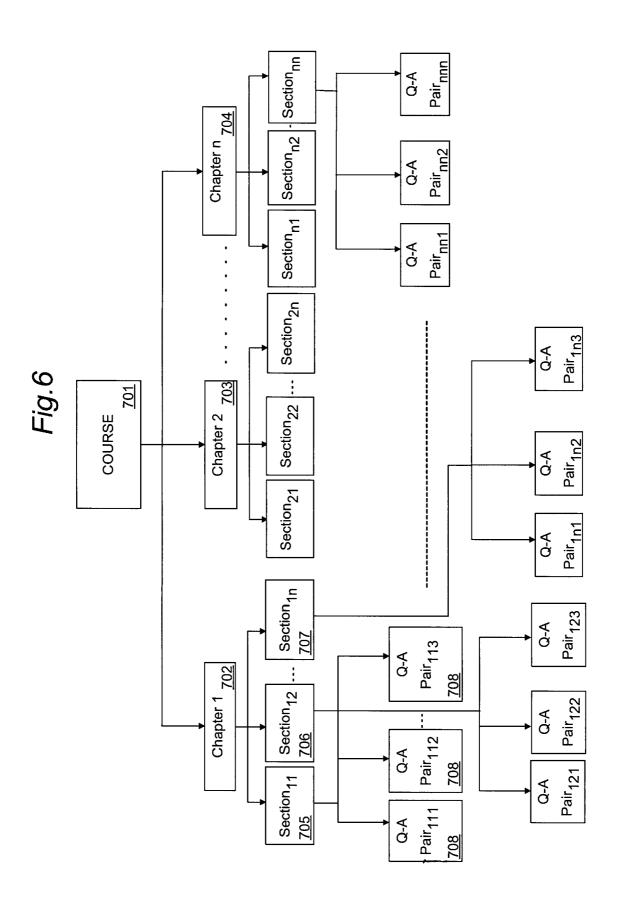


U.S. Patent

Dec. 16, 2003

Sheet 13 of 31

US 6,665,640 B1



Dec. 16, 2003

Sheet 14 of 31

FIELD NAME	DATA TYPE	Size	Norr	PRIMARY KEY	INDEXED?
701A	702A	<u>703A</u>	704A	705A	706A
ChapterName 707A	Varchar	255	No	°N	Yes
SectionName	Varchar	255	°Z	^S	Yes
708A					-

Fig.7B

FIELD NAME 720	DATA TYPE <u>721</u>	Size <u>722</u>	Null <u>723</u>	PRIMARY KEY 724	INDEXED? <u>725</u>
Chapter_ID <u>726</u>	Integer		No	Yes	Yes
Answer_ID 727	Char	5	No	UNIQUE	Yes
Section_Name 728	Varchar	255	No	UNIQUE	Yes
Answer_Title	Varchar	255	Yes	No	Yes
PairedQuestion 730	Text	16	No	No	Yes (Full-Text)
AnswerPath <u>731</u>	Varchar	255	No	No	Yes
Creator <u>732</u>	Varchar	50	No	No	Yes
Date_of_Creation 733	Date	-	No	No	Yes
Date_of_Modification 734	Date	-	No	No	Yes

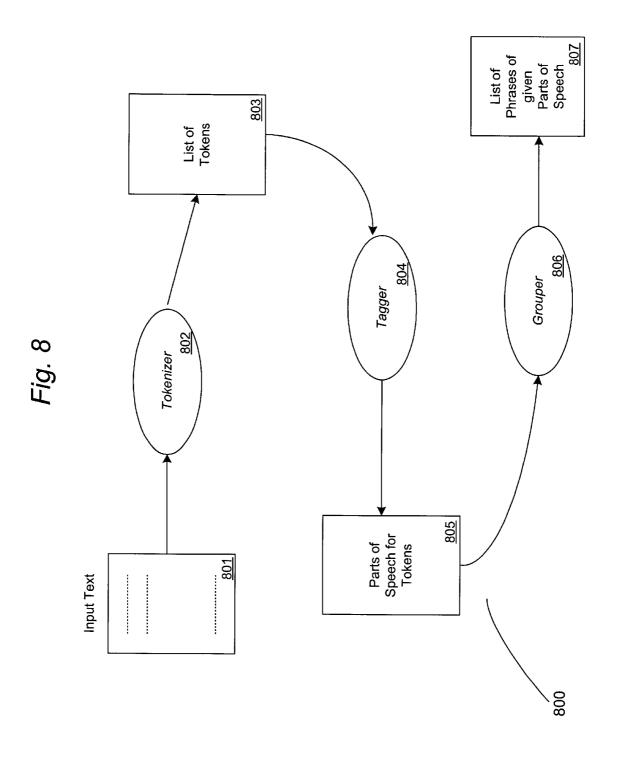
Fig. 7C

Field	<u>720</u>	Description	
AnswerID	<u>727</u>	An integer - automatically incremented for user convenience	
Section_Name	<u>728</u>	Name of section to which the particular record belongs. This field along with AnswerlD has to be made primary key	
Answer_Title	<u>729</u>	A short description of the answer	
PairedQuestion	<u>730</u>	Contains one or more combinations of questions for the related answer whose path is stored in the next column AnswerPath	
AnswerPath	<u>731</u>	Contains the path of text file, which contains theanswer to the rleated qusetions stored in the previous column	
Creator	<u>732</u>	Name of content creator	
Date_of_Creation	<u>733</u>	Date on which content has been added	
Date_of_Modification	<u>734</u>	Date on which content has been changed or modified	

Fig. 7D

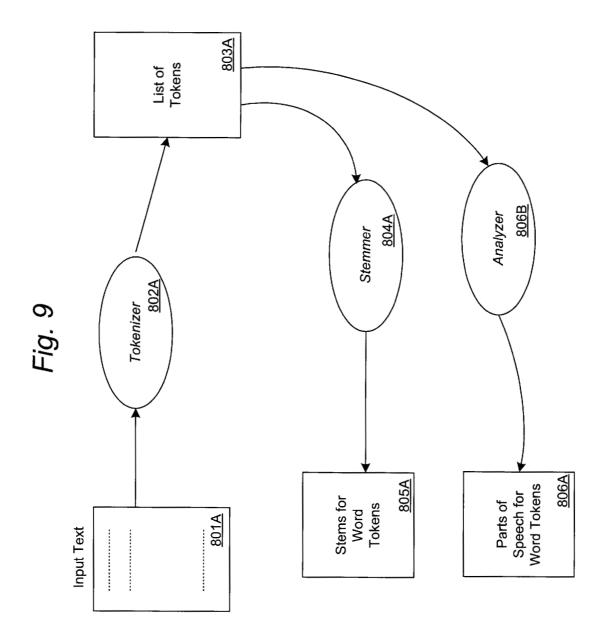
FIELD <u>740</u>	Dата Түре <u>741</u>	Size <u>742</u>	NULL <u>743</u>	PRIMARY KEY <u>744</u>	INDEXED 745
Answer_ID <u>746</u>	Char	5	No	Yes	Yes
Answer_Title 747	Varchar	255	Yes	No	No
PairedQuestion 748	Text	16	No	No	Yes (Full-Text)
Answer_Path 749	Varchar	255	No	No	No
Creator 750	Varchar	50	No	No	No
Date_of_Creation 751	Date	-	No	No	No
Date_of_Modification 752	Date	-	No	No	No

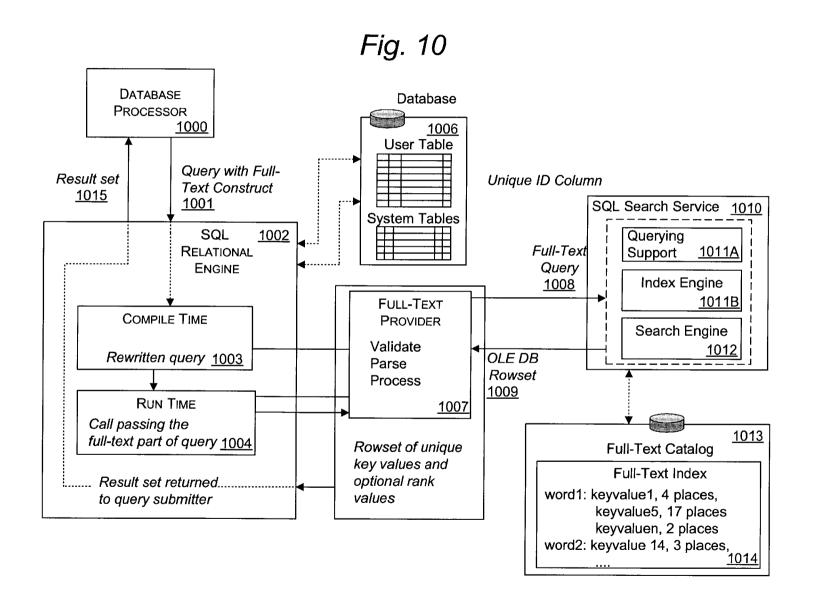
U.S. Patent Dec. 16, 2003 Sheet 18 of 31 US 6,665,640 B1



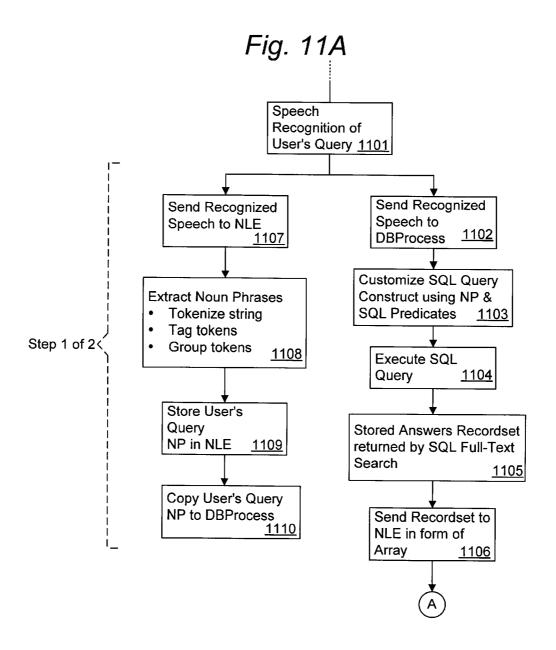
Dec. 16, 2003

Sheet 19 of 31



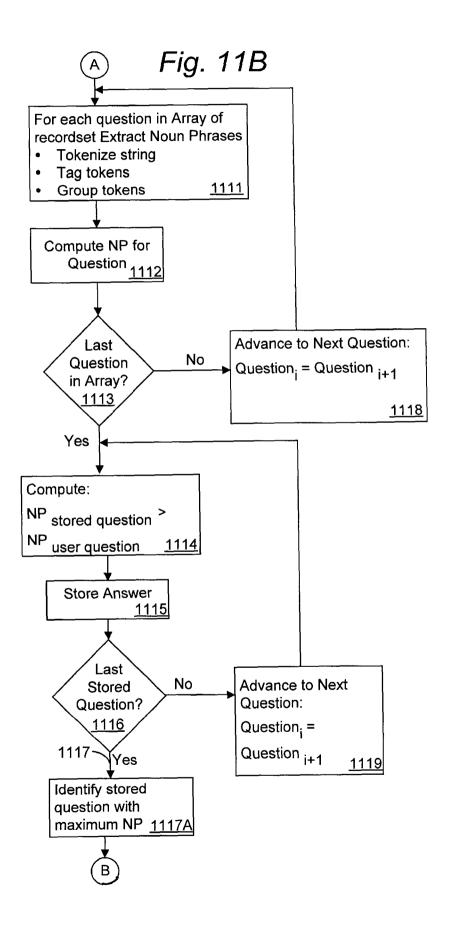


U.S. Patent Dec. 16, 2003 Sheet 21 of 31 US 6,665,640 B1



Dec. 16, 2003

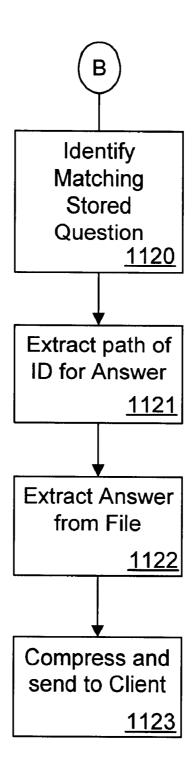
Sheet 22 of 31

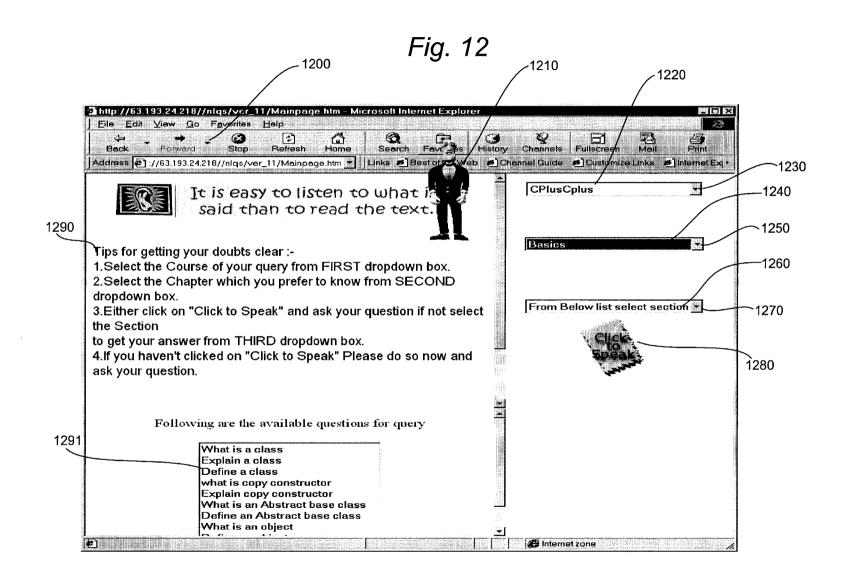


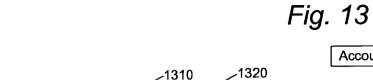
Dec. 16, 2003

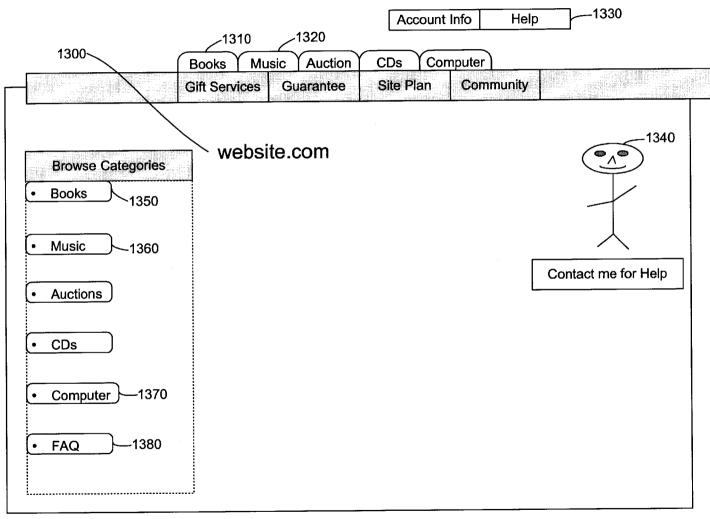
Sheet 23 of 31

Fig. 11C



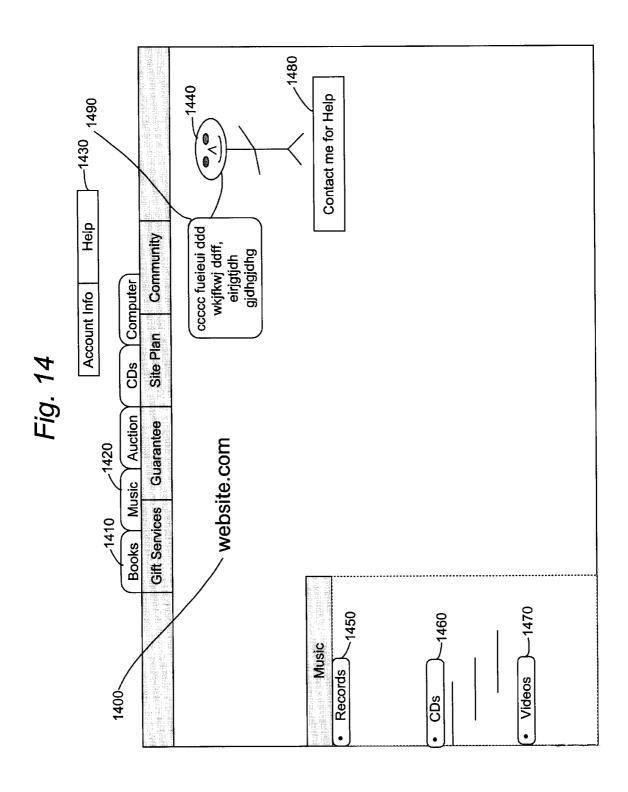






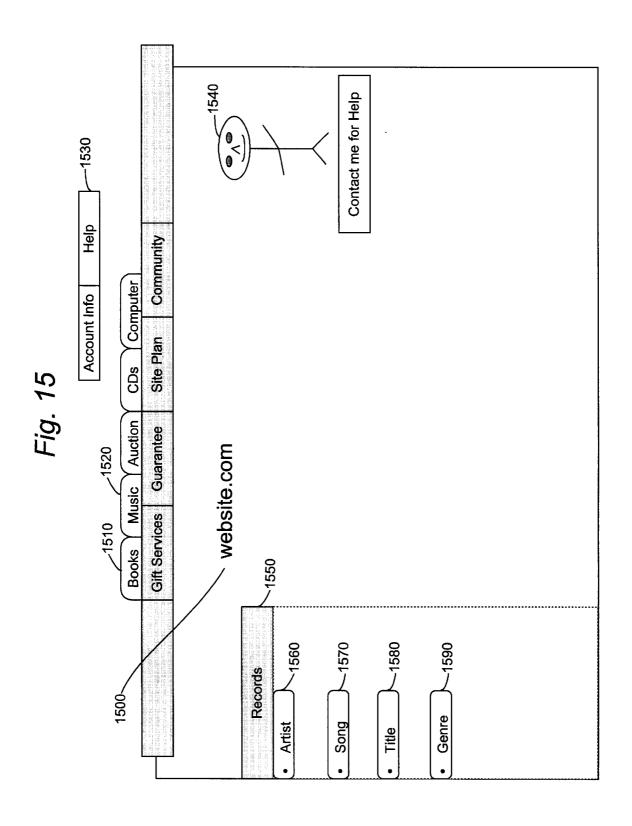
Dec. 16, 2003

Sheet 26 of 31



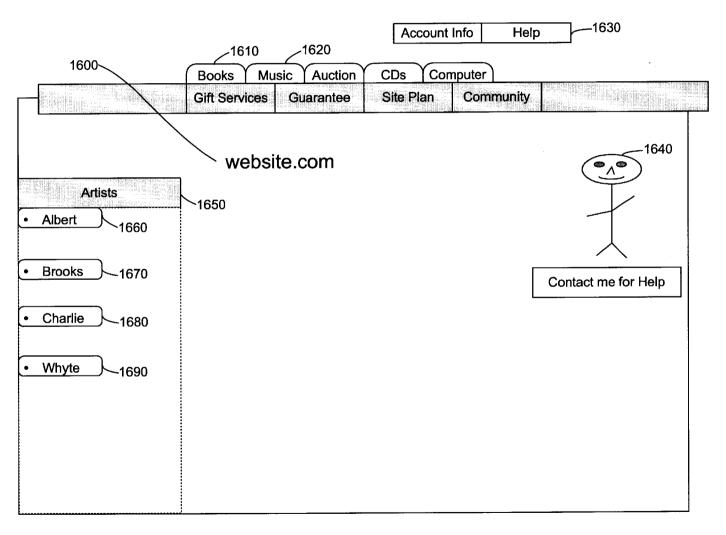
Dec. 16, 2003

Sheet 27 of 31



U.S. Patent

Fig. 16



Dec. 16, 2003

Sheet 29 of 31

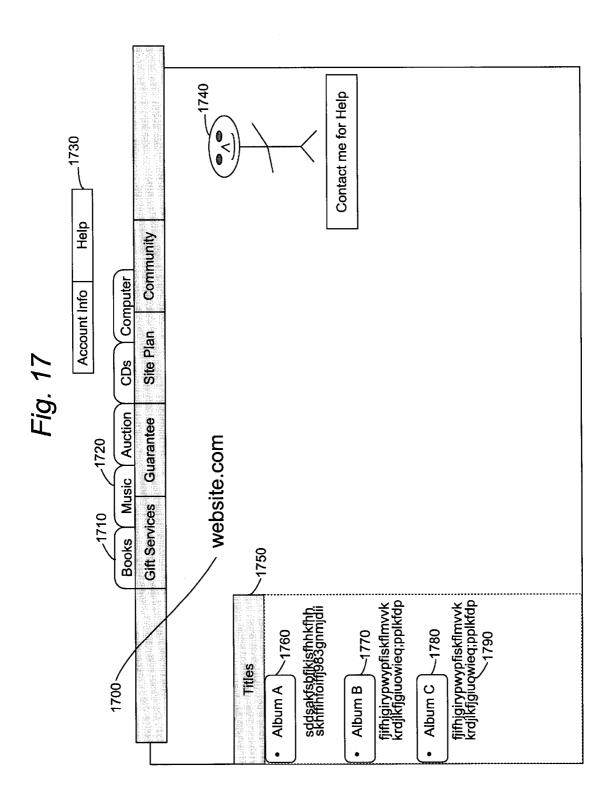


Fig. 18 (Page 1/2)

Dec. 16, 2003

Sheet 30 of 31

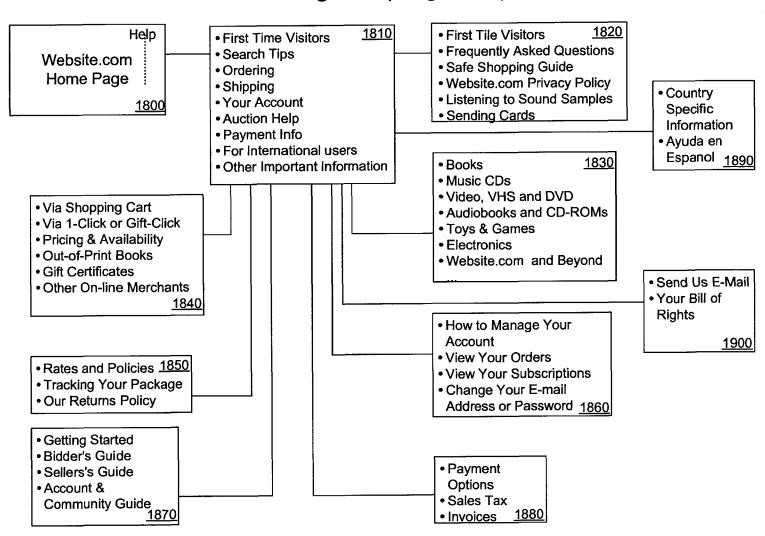
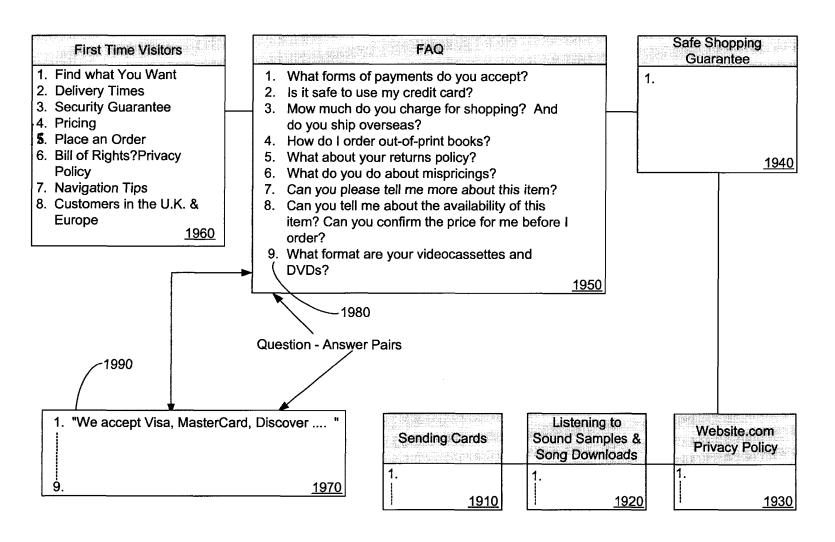


Fig. 18 (Page 2/2)



INTERACTIVE SPEECH BASED LEARNING/ TRAINING SYSTEM FORMULATING SEARCH QUERIES BASED ON NATURAL LANGUAGE PARSING OF RECOGNIZED **USER OUERIES**

RELATED APPLICATIONS

The present application is related to the following applications also filed contemporaneously herewith:

- 1) Ser. No. 09/439,145 entitled Distributed Real Time Speech Recognition System, attorney docket no. PHO
- 2) Ser. No. 09/439,174 entitled Internet Server with Speech Support for Enhanced Interactivity—attorney 15 docket no. PHO 99-003;
- 3) Ser. No. 09/439,060 entitled Intelligent Query Engine For Processing Voice Based Queries—attorney docket no. PHO 99-004;

The above are incorporated by reference herein.

FIELD OF THE INVENTION

The invention relates to a system and an interactive method for presenting an interactive, real-time speechenabled tutorial over a distributed network such as the INTERNET or local intranet. This interactive system is especially useful when implemented over the World-Wide Web services (WWW of the INTERNET, functions so that a user/student can learn and interact with an animated agent who answers speech-based queries in a real-time fashion, 30 thus providing a human-like dialog experience.

BACKGROUND OF THE INVENTION

The INTERNET, and in particular, the World-Wide Web (WWW, is growing in popularity and usage for both commercial and recreational purposes, and this trend is expected to continue. This phenomenon is being driven, in part, by the increasing and widespread use of personal computer systems and the availability of low cost INTERNET access.

The emergence of inexpensive INTERNET access devices and high speed access techniques such as ADSL, cable modems, satellite modems, and the like, are expected to further accelerate the mass usage of the WWW.

Accordingly, it is expected that the number of entities 45 offering services, products, etc., over the WWW will increase dramatically over the coming years. Until now, however, the INTERNET "experience" for users has been limited mostly to non-voice based input/output devices, such as keyboards, intelligent electronic pads, mice, trackballs, 50 dialog involves the use of oral feedback. In other words, printers, monitors, etc. This presents somewhat of a bottleneck for interacting over the WWW for a variety of reasons.

First, there is the issue of familiarity. Many kinds of applications lend themselves much more naturally and fluently to a voice-based environment. For instance, most 55 people shopping for audio recordings are very comfortable with asking alive sales clerk in a record store for information on titles by a particular author, where they can be found in the store, etc. While it is often possible to browse and search on one's own to locate items of interest, it is usually easier and more efficient to get some form of human assistance first, and, with few exceptions, this request for assistance is presented in the form of a oral query. In addition, many persons cannot or will not, because of physical or psychological barriers, use any of the aforementioned conventional I/O devices. For example, many older persons cannot easily read the text presented on WWW pages, or understand the

layout/hierarchy of menus, or manipulate a mouse to make finely coordinated movements to indicate their selections. Many others are intimidated by the look and complexity of computer systems, WWW pages, etc., and therefore do not attempt to use online services for this reason as well.

Thus, applications which can mimic normal human interactions are likely to be preferred by potential on-line shoppers and persons looking for information over the WWW. It is also expected that the use of voice-based systems will increase the universe of persons willing to engage in e-commerce, e-learning, etc. To date, however, there are very few systems, if any, which permit this type of interaction, and, if they do, it is very limited. For example, various commercial programs sold by IBM (VIAVOICETM) and Kurzweil (DRAGONTM) permit some user control of the interface (opening, closing files) and searching (by using previously trained URLs) but they do not present a flexible solution that can be used by a number of users across multiple cultures and without time consuming voice training. Typical prior efforts to implement voice based functionality in an INTERNET context can be seen in U.S. Pat. No. 5,819,220 incorporated by reference herein.

Another issue presented by the lack of voice-based systems is efficiency. Many companies are now offering technical support over the INTERNET, and some even offer live operator assistance for such queries. While this is very advantageous (for the reasons mentioned above) it is also extremely costly and inefficient, because a real person must be employed to handle such queries.

This presents a practical limit that results in long wait times for responses or high labor overheads. An example of this approach can be seen U.S. Pat. No. 5,802,526 also incorporated by reference herein. In general, a service presented over the WWW is far more desirable if it is 'scalable," or, in other words, able to handle an increasing amount of user traffic with little if any perceived delay or troubles by a prospective user.

In a similar context, while remote learning has become an $_{40}\,$ increasingly popular option for many students, it is practically impossible for an instructor to be able to field questions from more than one person at a time. Even then, such interaction usually takes place for only a limited period of time because of other instructor time constraints. To date, however, there is no practical way for students to continue a human-like question and answer type dialog after the learning session is over, or without the presence of the instructor to personally address such queries.

Conversely, another aspect of emulating a human-like many persons prefer to receive answers and information in audible form. While a form of this functionality is used by some websites to communicate information to visitors, it is not performed in a real-time, interactive question-answer dialog fashion so its effectiveness and usefulness is limited.

Yet another area that could benefit from speech-based interaction involves so-called "search" engines used by INTERNET users to locate information of interest at web sites, such as the those available at YAHOO®.com, METACRAWLER®.com, EXCITE®.com, etc. These tools permit the user to form a search query using either combinations of keywords or metacategories to search through a web page database containing text indices associated with one or more distinct web pages. After processing the user's request, therefore, the search engine returns a number of hits which correspond, generally, to URL pointers and text excerpts from the web pages that represent the closest match

3

made by such search engine for the particular user query based on the search processing logic used by search engine. The structure and operation of such prior art search engines, including the mechanism by which they build the web page database, and parse the search query, are well known in the art. To date, applicant is unaware of any such search engine that can easily and reliably search and retrieve information based on speech input from a user.

There are a number of reasons why the above environments (e-commerce, e-support, remote learning, INTER-NET searching, etc.) do not utilize speech-based interfaces, despite the many benefits that would otherwise flow from such capability. First, there is obviously a requirement that the output of the speech recognizer be as accurate as possible. One of the more reliable approaches to speech recognition used at this time is based on the Hidden Markov Model (HMN)—a model used to mathematically describe any time series. A conventional usage of this technique is disclosed, for example, in U.S. Pat. No. 4,587,670 incorporated by reference herein. Because speech is considered to have an underlying sequence of one or more symbols, the 20 HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of 25 probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density B_i(O_i). It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model. The basic theory of HMMs was published in a series of classic papers by Baum and his 35 colleagues in the late 1960's and early 1970's. HMMs were first used in speech applications by Baker at Carnegie Mellon, by Jelenik and colleagues at IBM in the late 1970's and by Steve Young and colleagues at Cambridge University, UK in the 1990's. Some typical papers and texts are as follows:

- 1. L. E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains". Ann. Math. Stat., 37:1554-1563,1966
- 2. L. E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1-8,
- 3. J. H. Baker, "The dragon system—An Overview", IEEE Trans. on ASSP Proc., ASSP-23(1): 24-29, February, 1975
- 4. F. Jeninek et al, "Continuous Speech Recognition: Statistical methods" in Handbook of Statistics, II, P. R. Kristnaiad, Ed. Amsterdam, The Netherlands, North-Holland, 1982
- 5. L. R. Bahl, F. Jeninek, R. L. Mercer, "A maximum likelihood approach to continuous speech recognition", IEEE Trans. Pattern Anal. Mach. Intell., PAMI-5: 179-190,1983
- 6. J. D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, N.J. 1980.
- 7. H. R. Rabiner and B. H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, 1993
- 8. H. R. Rabiner, "Digital Processing of Speech Signals", Prentice Hall, 1978

More recently research has progressed in extending HMM and combining HMMs with neural networks to speech recognition applications at various laboratories. The following is a representative paper:

9. Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco (1993), Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition. Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4 pp. 899-916.

Also in I. Guyon and P. Wang editors, Advances in Pattern Recognition Systems using Neural Networks, Vol. 7 of a Series in Machine Perception and Artificial Intelligence. World Scientific, February 1994.

All of the above are hereby incorporated by reference. While the HMM-based speech recognition yields very good results, contemporary variations of this technique cannot guarantee a word accuracy requirement of 100% exactly and consistently, as will be required for WWW applications for all possible all user and environment conditions. Thus, although speech recognition technology has been available for several years, and has improved significantly, the technical requirements have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily.

In contrast to word recognition, Natural language processing (NLP) is concerned with the parsing, understanding and indexing of transcribed utterances and larger linguistic units. Because spontaneous speech contains many surface phenomena such as disfluencies,—hesitations, repairs and restarts, discourse markers such as 'well' and other elements which cannot be handled by the typical speech recognizer, it is the problem and the source of the large gap that separates speech recognition and natural language processing technologies. Except for silence between utterances, another problem is the absence of any marked punctuation available for segmenting the speech input into meaningful units such as utterances. For optimal NLP performance, these types of phenomena should be annotated at its input. However, most continuous speech recognition systems produce only a raw sequence of words. Examples of conventional systems using NLP are shown in U.S. Pat. Nos. 4,991,094, 5,068,789, 5,146,405 and 5,680,628, all of which are incorporated by

Second, most of the very reliable voice recognition systems are speaker-dependent, requiring that the interface be "trained" with the user's voice, which takes a lot of time, and is thus very undesirable from the perspective of a WWW environment, where a user may interact only a few times with a particular website. Furthermore, speaker-dependent systems usually require a large user dictionary (one for each unique user) which reduces the speed of recognition. This makes it much harder to implement a real-time dialog interface with satisfactory response capability (i.e., something that mirrors normal conversation—on the order of 3-5 seconds is probably ideal). At present, the typical shrinkwrapped speech recognition application software include offerings from IBM (VIAVOICETM) and Dragon Systems (DRAGONTM). While most of these applications are adequate for dictation and other transcribing applications, they are woefully inadequate for applications such as NLQS where the word error rate must be close to 0%. In addition these offerings require long training times and are typically are non client-server configurations. Other types of trained systems are discussed in U.S. Pat. No. 5,231,670 assigned to Kurzweil, and which is also incorporated by reference herein.

Another significant problem faced in a distributed voicebased system is a lack of uniformity/control in the speech recognition process. In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. A well-known system of this type is depicted in U.S. Pat. No. 4,991,217 incorporated by reference herein. These clients can take numerous forms (desktop PC, laptop PC, PDA, etc.) having varying speech signal processing and communications capability. Thus, from the server side perspective, it is not easy to assure uniform treatment of all users accessing a voice-enabled web page, since such users may have significantly disparate word recognition and error rate performances. While a prior art reference to Gould et al.—U.S. Pat. No. 5,915,236—discusses generally the notion of tailoring a recognition process to a set of available computational resources, it does not address or attempt to solve the issue of how to optimize resources in a distributed environment such as a client-server model. Again, to enable such voice-based technologies on a wide-spread scale it is far more preferable to have a system that harmonizes and accounts for discrepancies in individual systems so that even 20 the thinnest client is supportable, and so that all users are able to interact in a satisfactory manner with the remote server running the e-commerce, e-support and/or remote learning application.

Two references that refer to a distributed approach for speech recognition include U.S. Pat. Nos. 5,956,683 and 5,960,399 incorporated by reference herein. In the first of these, U.S. Pat. 5,956,683—Distributed Voice Recognition System. (assigned to Qualcomm) an implementation of a distributed voice recognition system between a telephonybased handset and a remote station is described. In this implementation, all of the word recognition operations seem to take place at the handset. This is done since the patent describes the benefits that result from locating of the system for acoustic feature extraction at the portable or cellular phone in order to limit degradation of the acoustic features 35 due to quantization distortion resulting from the narrow bandwidth telephony channel. This reference therefore does not address the issue of how to ensure adequate performance for a very thin client platform. Moreover, it is difficult to determine, how, if at all, the system can perform real-time word recognition, and there is no meaningful description of how to integrate the system with a natural language processor.

The second of these references—U.S. Pat No. 5,960, 399—Client/Server Speech Processor/Recognizer (assigned to GTE) describes the implementation of a HMM-based distributed speech recognition system. This reference is not instructive in many respects, however, including how to optimize acoustic feature extraction for a variety of client platforms, such as by performing a partial word recognition process where appropriate. Most importantly, there is only a description of a primitive server-based recognizer that only recognizes the user's speech and simply returns certain keywords such as the user's name and travel destination to fill out a dedicated form on the user's machine. Also, the 55 streaming of the acoustic parameters does not appear to be implemented in real-time as it can only take place after silence is detected. Finally, while the reference mentions the possible use of natural language processing (column 9) there is no explanation of how such function might be implemented in a real-time fashion to provide an interactive feel for the user

SUMMARY OF THE INVENTION

an improved system and method for overcoming the limitations of the prior art noted above;

A primary object of the present invention is to provide a word and phrase recognition system that is flexibly and optimally distributed across a client/platform computing architecture, so that improved accuracy, speed and uniformity can be achieved for a wide group of users;

A further object of the present invention is to provide a speech recognition system that efficiently integrates a distributed word recognition system with a natural language processing system, so that both individual words and entire speech utterances can be quickly and accurately recognized in any number of possible languages;

A related object of the present invention is to provide an efficient query response system so that an extremely accurate, real-time set of appropriate answers can be given in response to speech-based queries;

Yet another object of the present invention is to provide an interactive, real-time instructional/learning system that is distributed across a client/server architecture, and permits a real-time question/answer session with an interactive character:

A related object of the present invention is to implement such interactive character with an articulated response capability so that the user experiences a human-like interaction;

Still a further object of the present invention is to provide an INTERNET website with speech processing capability so that voice based data and commands can be used to interact with such site, thus enabling voice-based e-commerce and e-support services to be easily scaleable;

Another object is to implement a distributed speech recognition system that utilizes environmental variables as part of the recognition process to improve accuracy and

A further object is to provide a scaleable query/response database system, to support any number of query topics and users as needed for a particular application and instantaneous demand;

Yet another object of the present invention is to provide a query recognition system that employs a two-step approach, including a relatively rapid first step to narrow down the list of potential responses to a smaller candidate set, and a second more computationally intensive second step to identify the best choice to be returned in response to the query from the candidate set;

A further object of the present invention is to provide a natural language processing system that facilitates query recognition by extracting lexical components of speech utterances, which components can be used for rapidly identifying a candidate set of potential responses appropriate for such speech utterances;

Another related object of the present invention is to provide a natural language processing system that facilitates query recognition by comparing lexical components of speech utterances with a candidate set of potential response to provide an extremely accurate best response to such query.

One general aspect of the present invention, therefore, relates to a natural language query system (NLQS) that offers a fully interactive method for answering user's questions over a distributed network such as the INTERNET or a local intranet. This interactive system when implemented over the worldwide web (WWW) services of the INTER-NET functions so that a client or user can ask a question in An object of the present invention, therefore, is to provide 65 a natural language such as English, French, German or Spanish and receive the appropriate answer at his or her personal computer also in his or her native natural language.

The system is distributed and consists of a set of integrated software modules at the client's machine and another set of integrated software programs resident on a server or set of servers. The client-side software program is comprised of a speech recognition program, an agent and its control program, and a communication program. The server-side program is comprised of a communication program, a natural language engine (NLE), a database processor (DBProcess), an interface program for interfacing the DBProcess with the NLE, and a SQL database. In addition, the client's machine is equipped with a microphone and a speaker. Processing of the speech utterance is divided between the client and server side so as to optimize processing and transmission latencies, and so as to provide support for even very thin client platforms.

In the context of an interactive learning application, the system is specifically used to provide a single-best answer to a user's question. The question that is asked at the client's machine is articulated by the speaker and captured by a microphone that is built in as in the case of a notebook computer or is supplied as a standard peripheral attachment. Once the question is captured, the question is processed partially by NLQS client-side software resident in the client's machine. The output of this partial processing is a set of speech vectors that are transported to the server via the INTERNET to complete the recognition of the user's questions. This recognized speech is then converted to text at the server.

After the user's question is decoded by the speech recognition engine (SRE) located at the server, the question is 30 converted to a structured query language (SQL) query. This query is then simultaneously presented to a software process within the server called DBProcess for preliminary processing and to a Natural Language Engine (NLE) module for extracting the noun phrases (NP) of the user's question. 35 During the process of extracting the noun phrase within the NLE, the tokens of the users' question are tagged. The tagged tokens are then grouped so that the NP list can be determined. This information is stored and sent to the DBProcess process.

In the DBProcess, the SQL query is fully customized using the NP extracted from the user's question and other environment variables that are relevant to the application. For example, in a training application, the user's selection of course, chapter and or section would constitute the environ- 45 ment variables. The SQL query is constructed using the extended SQL Full-Text predicates—CONTAINS, FREETEXT, NEAR, AND. The SQL query is next sent to the Full-Text search engine within the SQL database, where a Full-Text search procedure is initiated. The result of this 50 search procedure is recordset of answers. This recordset contains stored questions that are similar linguistically to the user's question. Each of these stored questions has a paired answer stored in a separate text file, whose path is stored in a table of the database.

The entire recordset of returned stored answers is then returned to the NLE engine in the form of an array. Each stored question of the array is then linguistically processed sequentially one by one. This linguistic processing constitutes the second step of a 2-step algorithm to determine the 60 single best answer to the user's question. This second step proceeds as follows: for each stored question that is returned in the recordset, a NP of the stored question is compared with the NP of the user's question. After all stored questions of the array are compared with the user's question, the stored question that yields the maximum match with the user's question is selected as the best possible stored question that

matches the user's question. The metric that is used to determine the best possible stored question is the number of noun phrases.

The stored answer that is paired to the best-stored question is selected as the one that answers the user's question. The ID tag of the question is then passed to the DBProcess. This DBProcess returns the answer which is stored in a file.

A communication link is again established to send the answer back to the client in compressed form. The answer once received by the client is decompressed and articulated to the user by the text-to-speech engine. Thus, the invention can be used in any number of different applications involving interactive learning systems, INTERNET related commerce sites, INTERNET search engines, etc.

Computer-assisted instruction environments often require the assistance of mentors or live teachers to answer questions from students. This assistance often takes the form of organizing a separate pre-arranged forum or meeting time that is set aside for chat sessions or live call-in sessions so that at a scheduled time answers to questions may be provided. Because of the time immediacy and the on-demand or asynchronous nature of on-line training where a student may log on and take instruction at any time and at any location, it is important that answers to questions be provided in a timely and cost-effective manner so that the user or student can derive the maximum benefit from the material presented.

This invention addresses the above issues. It provides the user or student with answers to questions that are normally channeled to a live teacher or mentor. This invention provides a single-best answer to questions asked by the student. The student asks the question in his or her own voice in the language of choice. The speech is recognized and the answer to the question is found using a number of technologies including distributed speech recognition, full-text search database processing, natural language processing and textto-speech technologies. The answer is presented to the user, as in the case of a live teacher, in an articulated manner by an agent that mimics the mentor or teacher, and in the language of choice—English, French, German, Japanese or other natural spoken language. The user can choose the agent's gender as well as several speech parameters such as pitch, volume and speed of the character's voice.

Other applications that benefit from NLQS are e-commerce applications. In this application, the user's query for a price of a book, compact disk or for the availability of any item that is to be purchased can be retrieved without the need to pick through various lists on successive web pages. Instead, the answer is provided directly to the user without any additional user input.

Similarly, it is envisioned that this system can be used to provide answers to frequently-asked questions (FAQs), and as a diagnostic service tool for e-support. These questions are typical of a give web site and are provided to help the user find information related to a payment procedure or the specifications of, or problems experienced with a product/ service. In all of these applications, the NLQS architecture can be applied.

A number of inventive methods associated with these architectures are also beneficially used in a variety of INTERNET related applications.

Although the inventions are described below in a set of preferred embodiments, it will be apparent to those skilled in the art the present inventions could be beneficially used in many environments where it is necessary to implement fast, accurate speech recognition, and/or to provide a human-like dialog capability to an intelligent system.

Document 52-3

BRIEF DESCRIPTION OF THE DRAWINGS

- FIG. 1 is a block diagram of a preferred embodiment of a natural language query system (NLQS) of the present invention, which is distributed across a client/server computing architecture, and can be used as an interactive learning system, an e-commerce system, an e-support system, and the like;
- FIG. 2 is a block diagram of a preferred embodiment of a client side system, including speech capturing modules, partial speech processing modules, encoding modules, transmission modules, agent control modules, and answer/voice feedback modules that can be used in the aforementioned
- FIG. 2—2 is a block diagram of a preferred embodiment 15 of a set of initialization routines and procedures used for the client side system of FIG. 2;
- FIG. 3 is a block diagram of a preferred embodiment of a set of routines and procedures used for handling an iterated set of speech utterances on the client side system of FIG. 2, 20 transmitting speech data for such utterances to a remote server, and receiving appropriate responses back from such server:
- FIG. 4 is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for 25 un-initializing the client side system of FIG. 2;
- FIG. 4A is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a distributed component of a speech recognition module for the server side system of FIG. 5;
- FIG. 4B is a block diagram of a preferred set of routines and procedures used for implementing an SQL query builder for the server side system of FIG. 5;
- FIG. 4C is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a database control process module for the server side system of FIG. 5;
- FIG. 4D is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a natural language engine that provides query formulation support, a query response module, and an interface to the database control process module for the server side system of FIG. 5;
- FIG. 5 is a block diagram of a preferred embodiment of a server side system, including a speech recognition module to complete processing of the speech utterances, environmental and grammar control modules, query formulation modules, a natural language engine, a database control module, and a query response module that can be used in the aforementioned NLQS;
- FIG. 6 illustrates the organization of a full-text database used as part of server side system shown in FIG. 5;
- FIG. 7A illustrates the organization of a full-text database course table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention:
- FIG. 7B illustrates the organization of a full-text database chapter table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present 60
- FIG. 7C describes the fields used in a chapter table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention;
- as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention;

10

- FIG. 8 is a flow diagram of a first set of operations performed by a preferred embodiment of a natural language engine on a speech utterance including Tokenization, Tagging and Grouping;
- FIG. 9 is a flow diagram of the operations performed by a preferred embodiment of a natural language engine on a speech utterance including stemming and Lexical Analysis
- FIG. 10 is a block diagram of a preferred embodiment of a SQL database search and support system for the present
- FIGS. 11A-11C are flow diagrams illustrating steps performed in a preferred two step process implemented for query recognition by the NLQS of FIG. 2;
- FIG. 12 is an illustration of another embodiment of the present invention implemented as part of a Web-based speech based learning/training System;
- FIGS. 13-17 are illustrations of another embodiment of the present invention implemented as part of a Web-based e-commerce system;
- FIG. 18 is an illustration of another embodiment of the present invention implemented as part of a voice-based Help Page for an E-Commerce Web Site.

DETAILED DESCRIPTION OF THE INVENTION

Overview

As alluded to above, the present inventions allow a user to ask a question in a natural language such as English, French, German, Spanish or Japanese at a client computing system (which can be as simple as a personal digital assistant or cell-phone, or as sophisticated as a high end desktop PC) and receive an appropriate answer from a remote server also in his or her native natural language. As such, the embodiment of the invention shown in FIG. 1 is beneficially used in what can be generally described as a Natural Language Query System (NLQS) 100, which is configured to interact on a real-time basis to give a human-like dialog capability/ experience for e-commerce, e-support, and e-learning applications.

The processing for NLQS 100 is generally distributed across a client side system 150, a data link 160, and a server-side system 180. These components are well known in the art, and in a preferred embodiment include a personal computer system 150, an INTERNET connection 160A, 160B, and a larger scale computing system 180. It will be understood by those skilled in the art that these are merely 50 exemplary components, and that the present invention is by no means limited to any particular implementation or combination of such systems. For example, client-side system 150 could also be implemented as a computer peripheral, a PDA, as part of a cell-phone, as part of an INTERNETadapted appliance, an INTERNET linked kiosk, etc. Similarly, while an INTERNET connection is depicted for data link 160A, it is apparent that any channel that is suitable for carrying data between client system 150 and server system 180 will suffice, including a wireless link, an RF link, an IR link, a LAN, and the like. Finally, it will be further appreciated that server system 180 may be a single, largescale system, or a collection of smaller systems interlinked to support a number of potential network users.

Initially speech input is provided in the form of a question FIG. 7D describes the fields used in a section table used 65 or query articulated by the speaker at the client's machine or personal accessory as a speech utterance. This speech utterance is captured and partially processed by NLQS client-

Document 52-3

11

side software 155 resident in the client's machine. To facilitate and enhance the human-like aspects of the interaction, the question is presented in the presence of an animated character 157 visible to the user who assists the user as a personal information retriever/agent. The agent can also interact with the user using both visible text output on a monitor/display (not shown) and/or in audible form using a text to speech engine 159. The output of the partial processing done by SRE 155 is a set of speech vectors that are transmitted over communication channel 160 that links the user's machine or personal accessory to a server or servers via the INTERNET or a wireless gateway that is linked to the INTERNET as explained above. At server 180, the partially processed speech signal data is handled by a server-side SRE 182, which then outputs recognized speech text corresponding to the user's question. Based on this user question related text, a text-to-query converter 184 formulates a suitable query that is used as input to a database processor 186. Based on the query, database processor 186 then locates and retrieves an appropriate answer using a customized SQL query from database 188. A Natural Lan- 20 guage Engine 190 facilitates structuring the query to database 188. After a matching answer to the user's question is found, the former is transmitted in text form across data link 160B, where it is converted into speech by text to speech engine 159, and thus expressed as oral feedback by animated 25 character agent 157.

Because the speech processing is broken up in this fashion, it is possible to achieve real-time, interactive, human-like dialog consisting of a large, controllable set of questions/answers. The assistance of the animated agent 157 further enhances the experience, making it more natural and comfortable for even novice users. To make the speech recognition process more reliable, context-specific grammars and dictionaries are used, as well as natural language processing routines at NLE 190, to analyze user questions lexically. While context-specific processing of speech data is known in the art (see e.g., U.S. Pat Nos. 5,960,394, 5,867, 817, 5,758,322 and 5,384,892 incorporated by reference herein) the present inventors are unaware of any such implementation as embodied in the present inventions. The $_{40}$ text of the user's question is compared against text of other questions to identify the question posed by the user by DB processor/engine (DBE) 186. By optimizing the interaction and relationship of the SR engines 155 and 182, the NLP routines 190, and the dictionaries and grammars, an $_{45}$ extremely fast and accurate match can be made, so that a unique and responsive answer can be provided to the user.

On the server side 180, interleaved processing further accelerates the speech recognition process. In simplified terms, the query is presented simultaneously both to NLE 50 190 after the query is formulated, as well as to DBE 186. NLE 190 and SRE 182 perform complementary functions in the overall recognition process. In general, SRE 182 is primarily responsible for determining the identity of the words articulated by the user, while NLE 190 is responsible 55 for the linguistic morphological analysis of both the user's query and the search results returned after the database query.

After the user's query is analyzed by NLE 190 some parameters are extracted and sent to the DBProcess. Additional statistics are stored in an array for the 2^{nd} step of processing. During the 2^{nd} step of 2-step algorithm, the recordset of preliminary search results are sent to the NLE 160 for processing. At the end of this 2^{nd} step, the single cess where further processing yields the paired answer that is paired with the single best stored question.

12

Thus, the present invention uses a form of natural language processing (NLP) to achieve optimal performance in a speech based web application system. While NLP is known in the art, prior efforts in Natural Language Processing (NLP) work nonetheless have not been well integrated with Speech Recognition (SR) technologies to achieve reasonable results in a web-based application environment. In speech recognition, the result is typically a lattice of possible recognized words each with some probability of fit with the speech recognizer. As described before, the input to a typical NLP system is typically a large linguistic unit. The NLP system is then charged with the parsing, understanding and indexing of this large linguistic unit or set of transcribed utterances. The result of this NLP process is to understand lexically or morphologically the entire linguistic unit as opposed to word recognition. Put another way, the linguistic unit or sentence of connected words output by the SRE has to be understood lexically, as opposed to just being "recognized"

As indicated earlier, although speech recognition technology has been available for several years, the technical requirements for the NLQS invention have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily. In realizing that even with the best of conditions, it might be not be possible to achieve the perfect 100% speech recognition accuracy that is required, the present invention employs an algorithm that balances the potential risk of the speech recognition process with the requirements of the natural language processing so that even in cases where perfect speech recognition accuracy is not achieved for each word in the query, the entire query itself is nonetheless recognized with sufficient accuracy.

This recognition accuracy is achieved even while meeting very stringent user constraints, such as short latency periods of 3 to 5 seconds (ideally-ignoring transmission latencies which can vary) for responding to a speech-based query, and for a potential set of 100-250 query questions. This quick response time gives the overall appearance and experience of a real-time discourse that is more natural and pleasant from the user's perspective. Of course, non-real time applications, such as translation services for example, can also benefit from the present teachings as well, since a centralized set of HMMs, grammars, dictionaries, etc., are maintained.

General Aspects of Speech Recognition Used in the Present Inventions

General background information on speech recognition can be found in the prior art references discussed above and incorporated by reference herein. Nonetheless, a discussion of some particular exemplary forms of speech recognition structures and techniques that are well-suited for NLQS 100 is provided next to better illustrate some of the characteristics, qualities and features of the present inven-

Speech recognition technology is typically of two types speaker independent and speaker dependent. In speakerdependent speech recognition technology, each user has a voice file in which a sample of each potentially recognized word is stored. Speaker-dependent speech recognition systems typically have large vocabularies and dictionaries making them suitable for applications as dictation and text question that matches the user's query is sent to the DBPro- 65 transcribing. It follows also that the memory and processor resource requirements for the speaker-dependent can be and are typically large and intensive.

13

Conversely speaker-independent speech recognition technology allows a large group of users to use a single vocabulary file. It follows then that the degree of accuracy that can be achieved is a function of the size and complexity of the grammars and dictionaries that can be supported for a given language. Given the context of applications for which NLQS, the use of small grammars and dictionaries allow speaker independent speech recognition technology to be implemented in NLQS.

The key issues or requirements for either type—speaker- 10 independent or speaker-dependent, are accuracy and speed. As the size of the user dictionaries increase, the speech recognition accuracy metric—word error rate (WER) and the speed of recognition decreases. This is so because the search time increases and the pronunciation match becomes 15 more complex as the size of the dictionary increases.

The basis of the NLQS speech recognition system is a series of Hidden Markov Models (HMN), which, as alluded to earlier, are mathematical models used to characterize any time varying signal. Because parts of speech are considered to be based on an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to an associated probability distribution. This finite state machine changes state once every time unit, and $\ ^{30}$ each time t such that a state j is entered, a spectral parameter vector O, is generated with probability density B_i(O_i). It is only the outcome, not the state which is visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.

In isolated speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word can each be described by a Markov model as follows:

$$O=o_1, o_2, \dots o_t$$
 (1-1)

where o, is a speech vector observed at time t. The isolated word recognition then is to compute:

$$\arg \max \{P(w_i|O)\} \tag{1-2}$$

By using Bayes' Rule,

$${P(w_i|O)} = [P(O|w_i)P(w_i)]/P(O)$$
(1-3)

In the general case, the Markov model when applied to speech also assumes a finite state machine which changes state once every time unit and each time that a state j is ⁵⁰ entered, a speech vector o_i is generated from the probability density $b_j(o_i)$. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij} .

For a state sequence X, the joint probability that O is 55 generated by the model M moving through a state sequence X is the product of the transition probabilities and the output probabilities. Only the observation sequence is known—the state sequence is hidden as mentioned before.

Given that X is unknown, the required likelihood is 60 computed by summing over all possible state sequences X=x(1), x(2), x(3), . . . x(t), that is

$$P(O|M) = \sum \{a_{x(0)x(1)} \prod b(x)(o_t) a_{x(t)x(t+1)}\}$$

Given a set of models M_i , corresponding to words w_i 65 equation 1-2 is solved by using 1-3 and also by assuming that:

14

 $P(O|w_i)=P(O|M_i)$

All of this assumes that the parameters $\{a_{ij}\}$ and $\{b_i(o_i)\}$ are known for each model M_i. This can be done, as explained earlier, by using a set of training examples corresponding to a particular model. Thereafter, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure. So if a sufficient number of representative examples of each word are collected, then a HMM can be constructed which simply models all of the many sources of variability inherent in real speech. This training is well-known in the art, so it is not described at length herein. except to note that the distributed architecture of the present invention enhances the quality of HMMs, since they are derived and constituted at the server side, rather than the client side. In this way, appropriate samples from users of different geographical areas can be easily compiled and analyzed to optimize the possible variations expected to be seen across a particular language to be recognized. Uniformity of the speech recognition process is also wellmaintained, and error diagnostics are simplified, since each prospective user is using the same set of HMMs during the recognition process.

To determine the parameters of a HMM from a set of training samples, the first step typically is to make a rough guess as to what they might be. Then a refinement is done using the Baum-Welch estimation formulae. By these formulae, the maximum likelihood estimates of μ_j (where μ_j is mean vector and Σ_i is covariance matrix) is:

$$\! = \! \mu_{j} \! = \! \boldsymbol{\Sigma}^{T}_{i-1} L_{j}(t) o_{i} / \! \big[\boldsymbol{\Sigma}^{T}_{i-1} L_{j}(t) o_{i} \big] \!$$

A forward-backward algorithm is next used to calculate the probability of state occupation $L_j(t)$. If the forward probability α_i (t) for some model M with N states is defined as:

$$\alpha_i(t)=P(o_1,\ldots,o_t,x(t)=j|M)$$

This probability can be calculated using the recursion:

$$\alpha_{i}(t) = [\sum_{i=2}^{N-1} \alpha(t-1)a_{ii}]b_{i}(o_{t})$$

Similarly the backward probability can be computed using the recursion:

$$\beta_i(t) = \sum_{i=2}^{N-1} a_{ii} b_i(o_{i+1})(t+1)$$

Realizing that the forward probability is a joint probability and the backward probability is a conditional probability, the probability of state occupation is the product of the two probabilities:

$$\alpha_i(t)\beta_i(t)=P(O, x(t)=j|M)$$

Hence the probability of being in state j at a time t is:

$$L_i(t)=1/P[\alpha_i(t)\beta_i(t)]$$

where P=P(O|M)

45

To generalize the above for continuous speech recognition, we assume the maximum likelihood state sequence where the summation is replaced by a maximum operation. Thus for a given model M, let ϕj (t) represent the maximum likelihood of observing speech vectors o_1 to o_r and being used in state j at time t:

$$\phi_i(t) = \max\{\phi_i(t)(t-1)\alpha_{ii}\}\cdot\beta_i(o_i)$$

Expressing this logarithmically to avoid underflow, this likelihood becomes:

15

 $\psi_i(t) = \max\{\psi_i(t-1) + \log(\alpha_{ij})\} + \log(b_i(o_i))$

This is also known as the Viterbi algorithm. It can be visualized as finding the best path through a matrix where the vertical dimension represents the states of the HMM and horizontal dimension represents frames of speech i.e. time. 5 To complete the extension to connected speech recognition, it is further assumed that each HMM representing the underlying sequence is connected. Thus the training data for continuous speech recognition should consist of connected utterances; however, the boundaries between words do not 10 have to be known.

To improve computational speed/efficiency, the Viterbi algorithm is sometimes extended to achieve convergence by using what is known as a Token Passing Model. The token passing model represents a partial match between the observation sequence o₁ to o_t and a particular model, subject to the constraint that the model is in state j at time t. This token passing model can be extended easily to connected speech environments as well if we allow the sequence of HMMs to be defined as a finite state network. A composite network that includes both phoneme-based HMMs and complete words can be constructed so that a single-best word can be recognized to form connected speech using word N-best extraction from the lattice of possibilities. This composite form of HMM-based connected speech recognizer is the 25 basis of the NLQS speech recognizer module. Nonetheless, the present invention is not limited as such to such specific forms of speech recognizers, and can employ other techniques for speech recognition if they are otherwise compatible with the present architecture and meet necessary per- 30 formance criteria for accuracy and speed to provide a real-time dialog experience for users.

The representation of speech for the present invention's HMM-based speech recognition system assumes that speech is essentially either a quasi-periodic pulse train (for voiced 35 speech sounds) or a random noise source (for unvoiced sounds). It may be modeled as two sources—one a impulse train generator with pitch period P and a random noise generator which is controlled by a voice/unvoiced switch. The output of the switch is then fed into a gain function 40 estimated from the speech signal and scaled to feed a digital filter H(z) controlled by the vocal tract parameter characteristics of the speech being produced. All of the parameters for this model—the voiced/unvoiced switching, the pitch period for voiced sounds, the gain parameter for the speech signal and the coefficient of the digital filter, vary slowly with time. In extracting the acoustic parameters from the user's speech input so that it can evaluated in light of a set of HMMs, cepstral analysis is typically used to separate the vocal tract information from the excitation information. The 50 cepstrum of a signal is computed by taking the Fourier (or similar) transform of the log spectrum. The principal advantage of extracting cepstral coefficients is that they are de-correlated and the diagonal covariances can be used with HMMs. Since the human ear resolves frequencies non- 55 linearly across the audio spectrum, it has been shown that a front-end that operates in a similar non-linear way improves speech recognition performance.

Accordingly, instead of a typical linear prediction-based analysis, the front-end of the NLQS speech recognition 60 engine implements a simple, fast Fourier transform based filter bank designed to give approximately equal resolution on the Mel-scale. To implement this filter bank, a window of speech data (for a particular time frame) is transformed using a software based Fourier transform and the magnitude 65 taken. Each FFT magnitude is then multiplied by the corresponding filter gain and the results accumulated. The

16

cepstral coefficients that are derived from this filter-bank analysis at the front end are calculated during a first partial processing phase of the speech signal by using a Discrete Cosine Transform of the log filter bank amplitudes. These cepstral coefficients are called Mel-Frequency Cepstral Coefficients (MFCC) and they represent some of the speech parameters transferred from the client side to characterize the acoustic features of the user's speech signal. These parameters are chosen for a number of reasons, including the fact that they can be quickly and consistently derived even across systems of disparate capabilities (i.e., for everything from a low power PDA to a high powered desktop system), they give good discrimination, they lend themselves to a number of useful recognition related manipulations, and they are relatively small and compact in size so that they can be transported rapidly across even a relatively narrow band link. Thus, these parameters represent the least amount of information that can be used by a subsequent server side system to adequately and quickly complete the recognition process.

To augment the speech parameters an energy term in the form of the logarithm of the signal energy is added. Accordingly, RMS energy is added to the 12 MFCC's to make 13 coefficients. These coefficients together make up the partially processed speech data transmitted in compressed form from the user's client system to the remote server side.

The performance of the present speech recognition system is enhanced significantly by computing and adding time derivatives to the basic static MFCC parameters at the server side. These two other sets of coefficients—the delta and acceleration coefficients representing change in each of the 13 values from frame to frame (actually measured across several frames), are computed during a second partial speech signal processing phase to complete the initial processing of the speech signal, and are added to the original set of coefficients after the latter are received. These MFCCs together with the delta and acceleration coefficients constitute the observation vector O, mentioned above that is used for determining the appropriate HMM for the speech data.

The delta and acceleration coefficients are computed using the following regression formula:

$$d_t = \sum_{\theta=1}^{\theta} [c_{t+\theta} - c_{t-\theta}]/2\sum_{\theta=1}^{\theta} e^2$$

where d_t is a delta coefficient at time t computed in terms of the corresponding static coefficients:

$$d_i = [c_{i+\theta} - c_{i-\theta}]/2\theta$$

In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. In other words, both the first and second partial processing phases above are executed by the same DSP (or microprocessor) running a ROM or software code routine at the client's computing machine.

In contrast, because of several considerations, specifically—cost, technical performance, and client hardware uniformity, the present NLQS system uses a partitioned or distributed approach. While some processing occurs on the client side, the main speech recognition engine runs on a centrally located server or number of servers. More specifically, as noted earlier, capture of the speech signals, MFCC vector extraction and compression are implemented on the client's machine during a first partial processing phase. The routine is thus streamlined and simple enough to be implemented within a browser program (as a plug in module, or a downloadable applet for example) for maxi-

17

mum ease of use and utility. Accordingly, even very "thin" client platforms can be supported, which enables the use of the present system across a greater number of potential sites. The primary MFCCs are then transmitted to the server over the channel, which, for example, can include a dial-up INTERNET connection, a LAN connection, a wireless connection and the like. After decompression, the delta and acceleration coefficients are computed at the server to complete the initial speech processing phase, and the resulting observation vectors O_t are also determined.

General Aspects of Speech Recognition Engine

The speech recognition engine is also located on the server, and is based on a HTK based recognition network compiled from a word-level network, a dictionary and a set of HMMs. The recognition network consists of a set of nodes connected by arcs. Each node is either a HMM model instance or a word end. Each model node is itself a network consisting of states connected by arcs. Thus when fully compiled, a speech recognition network consists of HMM states connected by transitions. For an unknown input utterance with T frames, every path from the start node to the exit node of the network passes through T HMM states. Each of these paths has log probability which is computed by summing the log probability of each individual transition in the path and the log probability of each emitting state generating the corresponding observation. The function of the Viterbi decoder is find those paths through the network which have the highest log probability. This is found using the Token Passing algorithm. In a network that has many nodes, the computation time is reduced by only allowing propagation of those tokens which will have some chance of becoming winners. This process is called pruning.

Natural Language Processor

In a typical natural language interface to a database, the user enters a question in his/her natural language, for example, English. The system parses it and translates it to a query language expression. The system then uses the query language expression to process the query and if the search is successful, a recordset representing the results is displayed in English either formatted as raw text or in a graphical form. For a natural language interface to work well involves a number of technical requirements.

For example, it needs to be robust—in the sentence 'What's the departments turnover' it needs to decide that the word whats=what's=what is. And it also has to determine that departments=department's. In addition to being robust, the natural language interface has to distinguish between the several possible forms of ambiguity that may exist in the natural language—lexical, structural, reference and ellipsis ambiguity. All of these requirements, in addition to the general ability to perform basic linguistic morphological operations of tokenization, tagging and grouping, are implemented within the present invention.

Tokenization is implemented by a text analyzer which treats the text as a series of tokens or useful meaningful units that are larger than individual characters, but smaller than phrases and sentences. These include words, separable parts 60 of words, and punctuation. Each token is associated with an offset and a length. The first phase of tokenization is the process of segmentation which extracts the individual tokens from the input text and keeps track of the offset where each token originated in the input text. The tokenizer output lists 65 the offset and category for each token. In the next phase of the text analysis, the tagger uses a built-in morphological

18

analyzer to look up each word/token in a phrase or sentence and internally lists all parts of speech. The output is the input string with each token tagged with a parts of speech notation. Finally the grouper which functions as a phrase extractor or phrase analyzer, determines which groups of words form phrases. These three operations which are the foundations for any modern linguistic processing schemes, are fully implemented in optimized algorithms for determining the single-best possible answer to the user's question.

SQL Database and Full-Text Query

Another key component of present system is a SQL-database. This database is used to store text, specifically the answer-question pairs are stored in full-text tables of the database. Additionally, the full-text search capability of the database allows full-text searches to be carried out.

While a large portion of all digitally stored information is in the form of unstructured data, primarily text, it is now possible to store this textual data in traditional database systems in character-based columns such as varchar and text. In order to effectively retrieve textual data from the database, techniques have to be implemented to issue queries against textual data and to retrieve the answers in a meaningful way where it provides the answers as in the case of the NLQS system.

There are two major types of textual searches: Property—This search technology first applies filters to documents in order to extract properties such as author, subject, type, word count, printed page count, and time last written, and then issues searches against those properties; Full-text—this search technology first creates indexes of all non-noise words in the documents, and then uses these indexes to support linguistic searches and proximity searches.

Two additional technologies are also implemented in this particular RDBMs: SQL Server also have been integrated: A Search service—a full-text indexing and search service that is called both index engine and search, and a parser that accepts full-text SQL extensions and maps them into a form that can be processed by the search engine.

The four major aspects involved in implementing full-text retrieval of plain-text data from a full-text-capable database are: Managing the definition of the tables and columns that are registered for full-text searches; Indexing the data in registered columns—the indexing process scans the character streams, determines the word boundaries (this is called word breaking), removes all noise words (this also is called stop words), and then populates a full-text index with the remaining words; Issuing queries against registered columns for populated full-text indexes; Ensuring that subsequent changes to the data in registered columns gets propagated to the index engine to keep the full-text indexes synchronized.

The underlying design principle for the indexing, querying, and synchronizing processes is the presence of a full-text unique key column (or single-column primary key) on all tables registered for full-text searches. The full-text index contains an entry for the non-noise words in each row together with the value of the key column for each row.

When processing a full-text search, the search engine returns to the database the key values of the rows that match the search criteria.

The full-text administration process starts by designating a table and its columns of interest for full-text search. Customized NLQS stored procedures are used first to register tables and columns as eligible for full-text search. After that, a separate request by means of a stored procedure is issued to populate the full-text indexes. The result is that the

19

underlying index engine gets invoked and asynchronous index population begins. Full-text indexing tracks which significant words are used and where they are located. For example, a full-text index might indicate that the word "NLQS" is found at word number 423 and word number 982 in the Abstract column of the DevTools table for the row associated with a ProductID of 6. This index structure supports an efficient search for all items containing indexed words as well as advanced search operations, such as phrase searches and proximity searches. (An example of a phrase 10 search is looking for "white elephant," where "white" is followed by "elephant". An example of a proximity search is looking for "big" and "house" where "big" occurs near "house".) To prevent the full-text index from becoming bloated, noise words such as "a," "and," and "the" are 15 ignored.

Extensions to the Transact-SQL language are used to construct full-text queries. The two key predicates that are used in the NLQS are CONTAINS and FREETEXT.

The CONTAINS predicate is used to determine whether or not values in full-text registered columns contain certain words and phrases. Specifically, this predicate is used to search for:

A word or phrase.

The prefix of a word or phrase.

A word or phrase that is near another.

A word that is an inflectional form of another (for example, "drive" is the inflectional stem of "drives," "drove," "driving," and "driven").

A set of words or phrases, each of which is assigned a different weighting.

The relational engine within SQL Server recognizes the CONTAINS and FREETEXT predicates and performs some minimal syntax and semantic checking, such as ensuring that the column referenced in the predicate has been registered for full-text searches. During query execution, a full-text predicate and other relevant information are passed to the full-text search component. After further syntax and semantic validation, the search engine is invoked and returns the set of unique key values identifying those rows in the table that satisfy the full-text search condition. In addition to the FREETEXT and CONTAINS, other predicates such as AND, LIKE, NEAR are combined to create the customized NLQS SQL construct.

Full-Text Query Architecture of the SQL Database

The full-text query architecture is comprised of the following several components—Full-Text Query component, the SQL Server Relational Engine, the Full-Text provider and the Search Engine.

The Full-Text Query component of the SQL database accept a full-text predicate or rowset-valued function from the SQL Server; transform parts of the predicate into an 55 internal format, and sends it to Search Service, which returns the matches in a rowset. The rowset is then sent back to SQL Server. SQL Server uses this information to create the resultset that is then returned to the submitter of the query.

The SQL Server Relational Engine accepts the CONTAINS and FREETEXT predicates as well as the CONTAINSTABLE() and FREETEXTTABLE() rowset-valued functions. During parse time, this code checks for conditions such as attempting to query a column that has not been registered for full-text search. If valid, then at run time, the ft_search_condition and context information is sent to the full-text provider. Eventually, the full-text provider

20

returns a rowset to SQL Server, which is used in any joins (specified or implied) in the original query. The Full-Text Provider parses and validates the ft_search_condition, constructs the appropriate internal representation of the full-text search condition, and then passes it to the search engine. The result is returned to the relational engine by means of a rowset of rows that satisfy ft_search_condition.

Client Side System 150

The architecture of client-side system 150 of Natural Language Query System 100 is illustrated in greater detail in FIG. 2. Referring to FIG. 2, the three main processes effectuated by Client System 150 are illustrated as follows: Initialization process 200A consisting of SRE 201, Communication 202 and Microsoft (NS) Agent 203 routines; an iterative process 200B consisting of two sub-routines: a) Receive User Speech 208-made up of SRE 204 and Communication 205; and b) Receive Answer from Server 207—made up of MS Speak Agent 206, Communication 209, Voice data file 210 and Text to Speech Engine 211. Finally, un-initialization process 200C is made up of three sub-routines: SRE 212, Communication 213, and MS Agent 214. Each of the above three processes are described in detail in the following paragraphs. It will be appreciated by those skilled in the art that the particular implementation for such processes and routines will vary from client platform to platform, so that in some environments such processes may be embodied in hard-coded routines executed by a dedicated DSP, while in others they may be embodied as software routines executed by a shared host processor, and in still others a combination of the two may be used.

Initialization at Client System 150

The initialization of the Client System 150 is illustrated in FIG. 2—2 and is comprised generally of 3 separate initializing processes: client-side Speech Recognition Engine 220A, MS Agent 220B and Communication processes 220C.

Initialization of Speech Recognition Engine 220A

Speech Recognition Engine 155 is initialized and configured using the routines shown in 220A. First, an SRE COM Library is initialized. Next, memory 220 is allocated to hold Source and Coder objects, are created by a routine 221. Loading of configuration file 221A from configuration data file 221B also takes place at the same time that the SRE Library is initialized. In configuration file 221B, the type of the input of Coder and the type of the output of the Coder are declared. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches. Accordingly, they are not discussed in detail herein. Next, Speech and Silence components of an utterance are calibrated using a routine 222, in a procedure that is also well-known in the art. To calibrate the speech and silence components, the user preferably articulates a sentence that is displayed in a text box on the screen. The SRE library then estimates the noise and other parameters required to find e silence and speech elements of future user utterances.

Initialization of MS Agent 220B

The software code used to initialize and set up a MS Agent 220B is also illustrated in FIG. 2—2. The MS Agent 220B routine is responsible for coordinating and handling the actions of the animated agent 157 (FIG. 1). This initialization thus consists of the following steps:

Document 52-3

21

- 1. Initialize COM library 223. This part of the code initializes the COM library, which is required to use ActiveX Controls, which controls are well-known in
- 2. Create instance of Agent Server 224—this part of the 5 code creates an instance of Agent ActiveX control.
- 3. Loading of MS Agent 225—this part of the code loads MS Agent character from a specified file 225A containing general parameter data for the Agent Character, such as the overall appearance, shape, size, etc.
- 4. Get Character Interface 226—this part of the code gets an appropriate interface for the specified character; for example, characters may have different control/ interaction capabilities that can be presented to the user.
- 5. Add Commands to Agent Character Option 227—this part of the code adds commands to an Agent Properties sheet, which sheet can be accessed by clicking on the icon that appears in the system tray, when the Agent character is loaded e.g., that the character can Speak, how he/she moves, TTS Properties, etc.
- 6. Show the Agent Character 228—this part of the code displays the Agent character on the screen so it can be seen by the user;
- 7. AgentNotifySink—to handle events. This part of the 25 code creates AgentNotifySink object 229, registers it at 230 and then gets the Agent Properties interface 231. The property sheet for the Agent character is assigned using routine 232.
- 8. Do Character Animations 233—This part of the code 30 plays specified character animations to welcome the user to NLQS 100.

The above then constitutes the entire sequence required to initialize the MS Agent. As with the SRE routines, the MS Agent routines can be implemented in any suitable and 35 conventional fashion by those skilled in the art based on the present teachings. The particular structure, operation, etc. of such routines is not critical, and thus they are not discussed in detail herein.

have an appearance and capabilities that are appropriate for the particular application. For instance, in a remote learning application, the agent has the visual form and mannerisms/ attitude/gestures of a college professor. Other visual props (blackboard, textbook, etc.) may be used by the agent and presented to the user to bring to mind the experience of being in an actual educational environment. The characteristics of the agent may be configured at the client side 150, and/or as part of code executed by a browser program (not from a particular web page. For example, a particular website offering medical services may prefer to use a visual image of a doctor. These and many other variations will be apparent to those skilled in the art for enhancing the humanlike, real-time dialog experience for users.

Initialization of Communication Link 160A

The initialization of Communication Link 160A is shown with reference to process 220C FIG. 2—2. Referring to FIG. 2—2, this initialization consists of the following code components: Open INTERNET Connection 234—this part of the code opens an INTERNET Connection and sets the parameter for the connection. Then Set Callback Status routine 235 sets the callback status so as to inform the user of the status of connection. Finally Start New HTTP INTERNET Session 65 236 starts a new INTERNET session. The details of Communications Link 160 and the set up process 220C are not

22 critical, and will vary from platform to platform. Again, in

some cases, users may use a low-speed dial-up connection, a dedicated high speed switched connection (T1 for example), an always-on xDSL connection, a wireless connection, and the like.

Iterative Processing of Queries/Answers

As illustrated in FIG. 3, once initialization is complete, an iterative query/answer process is launched when the user presses the Start Button to initiate a query. Referring to FIG. 3, the iterative query/answer process consists of two main sub-processes implemented as routines on the client side system 150: Receive User Speech 240 and Receive User Answer 243. The Receive User Speech 240 routine receives speech from the user (or another audio input source), while the Receive User Answer 243 routine receives an answer to the user's question in the form of text from the server so that it can be converted to speech for the user by text-to-speech engine 159. As used herein, the term "query" is referred to 20 in the broadest sense to refer, to either a question, a command, or some form of input used as a control variable by the system. For example, a, query may consist of a question directed to a particular topic, such as "what is a network" in the context of a remote learning application. In an e-commerce application a query might consist of a command to "list all books by Mark Twain" for example. Similarly, while the answer in a remote learning application consists of text that is rendered into audible form by the text to speech engine 159, it could also be returned as another form of multi-media information, such as a graphic image, a sound file, a video file, etc. depending on the requirements of the particular application. Again, given the present teachings concerning the necessary structure, operation, functions, performance, etc., of the client-side Receive User Speech 240 and Receiver User Answer 243 routines, one of ordinary skill in the art could implement such in a variety of wavs.

Receive User Speech—As illustrated in FIG. 3, the Receive User Speech routine 240 consists of a SRE 241 and In a preferred embodiment, the MS Agent is configured to 40 a Communication 242 process, both implemented again as routines on the client side system 150 for receiving and partially processing the user's utterance. SRE routine 241 uses a coder 248 which is prepared so that a coder object receives speech data from a source object. Next the Start Source 249 routine is initiated. This part of the code initiates data retrieval using the source Object which will in turn be given to the Coder object. Next, MFCC vectors 250 are extracted from the Speech utterance continuously until silence is detected. As alluded to earlier, this represents the shown) in response to configuration data and commands 50 first phase of processing of the input speech signal, and in a preferred embodiment, it is intentionally restricted to merely computing the MFCC vectors for the reasons already expressed above. These vectors include the 12 cepstral coefficients and the RMS energy term, for a total of 13 separate numerical values for the partially processed speech

In some environments, nonetheless, it is conceivable that the MFCC delta parameters and MFCC acceleration parameters can also be computed at client side system 150, depending on the computation resources available, the transmission bandwidth in data link 160A available to server side system 180, the speed of a transceiver used for carrying data in the data link, etc. These parameters can be determined automatically by client side system upon initializing SRE 155 (using some type of calibration routine to measure resources), or by direct user control, so that the partitioning of signal processing responsibilities can be optimized on a

23

case-by-case basis. In some applications, too, server side system 180 may lack the appropriate resources or routines for completing the processing of the speech input signal. Therefore, for some applications, the allocation of signal processing responsibilities may be partitioned differently, to the point where in fact both phases of the speech signal processing may take place at client side system 150 so that the speech signal is completely—rather than partially—processed and transmitted for conversion into a query at server side system 180.

Again in a preferred embodiment, to ensure reasonable accuracy and real-time performance from a query/response perspective, sufficient resources are made available in a client side system so that 100 frames per second of speech data can be partially processed and transmitted through link 15 160A. Since the least amount of information that is necessary to complete the speech recognition process (only 13 coefficients) is sent, the system achieves a real-time performance that is believed to be highly optimized, because other latencies (i.e., client-side computational latencies, packet formation latencies, transmission latencies) are minimized. It will be apparent that the principles of the present invention can be extended to other SR applications where some other methodology is used for breaking down the speech input signal by an SRE (i.e., non-MFCC based). The only criteria is that the SR processing be similarly dividable into multiple phases, and with the responsibility for different phases being handled on opposite sides of link 160A depending on overall system performance goals, requirements and the like. This functionality of the present invention can thus be achieved on a system-by-system basis, with an expected and typical amount of optimization being necessary for each particular implementation.

Thus, the present invention achieves a response rate performance that is tailored in accordance with the amount of information that is computed, coded and transmitted by the client side system 150. So in applications where real-time performance is most critical, the least possible amount of extracted speech data is transmitted to reduce these latencies, and, in other applications, the amount of extracted speech data that is processed, coded and transmitted can be varied.

Communication—transmit communication module 242 is used to implement the transport of data from the client to the server over the data link 160A, which in a preferred embodiment is the INTERNET. As explained above, the data consists of encoded MFCC vectors that will be used at then server-side of the Speech Recognition engine to complete the speech recognition decoding. The sequence of the communication is as follows:

OpenHTTPRequest 251—this part of the code first converts MFCC vectors to a stream of bytes, and then processes the bytes so that it is compatible with a protocol known as HTTP. This protocol is well-known in the art, and it is apparent that for other data links another suitable protocol 55 would be used.

- 1. Encode MFCC Byte Stream **251**—this part of the code encodes the MFCC vectors, so that they can be sent to the server via HTTP.
- 2. Send data **252**—this part of the code sends MFCC vectors 60 to the server using the INTERNET connection and the HTTP protocol.

Wait for the Server Response 253—this part of the code monitors the data link 160A a response from server side system 180 arrives. In summary, the MFCC parameters are 65 extracted or observed on-the-fly from the input speech signal. They are then encoded to a HTTP byte stream and

24

sent in a streaming fashion to the server before the silence is detected—i.e. sent to server side system 180 before the utterance is complete. This aspect of the invention also facilitates a real-time behavior, since data can be transmitted and processed even while the user is still speaking.

Receive Answer from Server 243 is comprised of the following modules as shown in FIG. 3.: MS Agent 244, Text-to-Speech Engine 245 and receive communication modules 246. All three modules interact to receive the answer from server side system 180. As illustrated in FIG. 3, the receive communication process consists of three separate processes implemented as a receive routine on client side system 150: a Receive the Best Answer 258 receives the best answer over data link 160B (the HTTP communication channel). The answer is de-compressed at 259 and then the answer is passed by code 260 to the MS Agent 244, where it is received by code portion 254. A routine 255 then articulates the answer using text-to-speech engine 257. Of course, the text can also be displayed for additional feedback purposes on a monitor used with client side system 150. The text to speech engine uses a natural language voice data file 256 associated with it that is appropriate for the particular language application (i.e., English, French, German, Japanese, etc.). As explained earlier when the answer is something more than text, it can be treated as desired to provide responsive information to the user, such as with a graphics image, a sound, a video clip,

Uninitialization

The un-initialization routines and processes are illustrated in FIG. 4. Three functional modules are used for un-initializing the primary components of the client side system 150; these include SRE 270, Communications 271 and MS Agent 272 un-initializing routines. To un-initialize SRE 220A, memory that was allocated in the initialization phase is de-allocated by code 273 and objects created during such initialization phase are deleted by code 274. Similarly, as illustrated in FIG. 4, to un-initialize Communications module 220C the INTERNET connection previously established with the server is closed by code portion 275 of the Communication Un-initialization routine 271. Next the INTERNET session created at the time of initialization is also closed by routine 276. For the un-initialization of the MS Agent 220B, as illustrated in FIG. 4, MS Agent Un-initialization routine 272 first releases the Commands Interface 227 using routine 277. This releases the commands added to the property sheet during loading of the agent character by routine 225. Next the Character Interface initialized by routine 226 is released by routine 278 and the Agent is unloaded at 279. The Sink Object Interface is then also released 280 followed by the release of the Property Sheet Interface 281. The Agent Notify Sink 282 then un-registers the Agent and finally the Agent Interface 283 is released which releases all the resources allocated during initialization steps identified in FIG. 2-2.

It will be appreciated by those skilled in the art that the particular implementation for such un-initialization processes and routines in FIG. 4 will vary from client platform to client platform, as for the other routines discussed above. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches without undue effort. Accordingly, they are not discussed in detail herein.

DESCRIPTION OF SERVER SIDE SYSTEM 180

Introduction

A high level flow diagram of the set of preferred processes implemented on server side system 180 of Natural Language

25

Query System 100 is illustrated in FIGS. 11A through FIG. 11C. In a preferred embodiment, this process consists of a two step algorithm for completing the processing of the speech input signal, recognizing the meaning of the user's query, and retrieving an appropriate answer/response for such query.

The 1st step as illustrated in FIG. 11A can be considered a high-speed first-cut pruning mechanism, and includes the following operations: after completing processing of the speech input signal, the user's query is recognized at step 1101, so that the text of the query is simultaneously sent to Natural Language Engine 190 (FIG. 1) at step 1107, and to DB Engine 186 (also FIG. 1) at step 1102. By "recognized" in this context it is meant that the user's query is converted into a text string of distinct native language words through $_{15}$ the HMM technique discussed earlier.

At NLE 190, the text string undergoes morphological linguistic processing at step 1108: the string is tokenized the tags are tagged and the tagged tokens are grouped Next the noun phrases (NP) of the string are stored at 1109, and also copied and transferred for use by DB Engine 186 during a DB Process at step 1110. As illustrated in FIG. 11A, the string corresponding to the user's query which was sent to the DB Engine 186 at 1102, is used together with the NP received from NLE 190 to construct an SQL Query at step 1103. Next, the SQL query is executed at step 1104, and a record set of potential questions corresponding to the user's query are received as a result of a full-text search at 1105, which are then sent back to NLE 190 in the form of an array at step 1106.

As can be seen from the above, this first step on the server side processing acts as an efficient and fast pruning mechanism so that the universe of potential "hits" corresponding to the user's actual query is narrowed down very quickly to a manageable set of likely candidates in a very short period of 35

Referring to FIG. 11B, in contrast to the first step above, the 2^{nd} step can be considered as the more precise selection portion of the recognition process. It begins with linguistic processing of each of the stored questions in the array 40 returned by the full-text search process as possible candidates representing the user's query. Processing of these stored questions continues in NLE 190 as follows: each question in the array of questions corresponding to the record set returned by the SQL full-text search undergoes morphological linguistic processing at step 1111: in this operation, a text string corresponding to the retrieved candidate question is tokenized, the tags are tagged and the tagged tokens are grouped. Next, noun phrases of the string are computed and stored at step 1112. This process continues 50 iteratively at point 1113, and the sequence of steps at 1118,1111, 1112, 1113 are repeated so that an NP for each retrieved candidate question is computed and stored. Once an NP is computed for each of the retrieved candidate questions of the array, a comparison is made between each 55 such retrieved candidate question and the user's query based on the magnitude of the NP value at step 1114. This process is also iterative in that steps 1114, 1115, 1116, 1119 are repeated so that the comparison of the NP for each retrieved candidate question with that of the NP of the user's query is completed. When there are no more stored questions in the array to be processed at step 1117, the stored question that has the maximum NP relative to the user's query, is identified at 1117A as the stored question which best matches the

Notably, it can be seen that the second step of the recognition process is much more computationally intensive 26

Filed 09/08/2008

than the first step above, because several text strings are tokenized, and a comparison is made of several NPs. This would not be practical, nonetheless, if it were not for the fact that the first step has already quickly and efficiently reduced the candidates to be evaluated to a significant degree. Thus, this more computationally intensive aspect of the present invention is extremely valuable, however because it yields extremely high accuracy in the overall query recognition process. In this regard, therefore, this second step of the query recognition helps to ensure the overall accuracy of the system, while the first step helps to maintain a satisfactory speed that provides a real-time feel for the user.

As illustrated in FIG. 11C, the last part of the query/ response process occurs by providing an appropriate matching answer/response to the user. Thus, an identity of a matching stored question is completed at step 1120. Next a file path corresponding to an answer of the identified matching question is extracted at step 1121. Processing continues so that the answer is extracted from the file path at 1122 and finally the answer is compressed and sent to client side system 150 at step 1123.

The discussion above is intended to convey a general overview of the primary components, operations, functions and characteristics of those portions of NLQS system 100 that reside on server side system 180. The discussion that follows describes in more detail the respective sub-systems.

Software Modules Used in Server Side System 180

The key software modules used on server-side system 180 of the NLQS system are illustrated in FIG. 5. These include generally the following components: a Communication module 500—identified as CommunicationServer ISAPI 500A (which is executed by SRE Server-side 182 —FIG. 1 and is explained in more detail below), and a database process DBProcess module 501 (executed by DB Engine **186**—FIG. 1). Natural language engine module **500**C (executed by NLE 190-FIG. 1) and an interface 500B between the NLE process module 500C and the DBProcess module 500B. As shown here, CommunicationServerISAPI 500A includes a server-side speech recognition engine and appropriate communication interfaces required between client side system 150 and server side system 180. As further illustrated in FIG. 5, server-side logic of Natural Language Query System 100 also can be characterized as including two dynamic link library components: CommunicationServerISAPI 500 and DBProcess 501. The CommunicationServerIASPI 500 is comprised of 3 sub-modules: Server-side Speech Recognition Engine module 500A; Interface module 500B between Natural Language Engine modules 500C and DBProcess 501; and the Natural Language Engine modules 500C.

DB Process 501 is a module whose primary function is to connect to a SOL database and to execute an SOL guery that is composed in response to the user's query. In addition, this module interfaces with logic that fetches the correct answer from a file path once this answer is passed to it from the Natural Language Engine module 500C.

Speech Recognition Sub-System 182 on Server-Side System 180

The server side speech recognition engine module 500A is a set of distributed components that perform the necessary functions and operations of speech recognition engine 182 (FIG. 1) at server-side 180. These components can be implemented as software routines that are executed by server side 180 in conventional fashion. Referring to FIG.

27

4A, a more detailed break out of the operation of the speech recognition components **600** at the server-side can be seen as follows:

Within a portion 601 of the server side SRE module 500A, the binary MFCC vector byte stream corresponding to the speech signal's acoustic features extracted at client side system 150 and sent over the communication channel 160 is received. The MFCC acoustic vectors are decoded from the encoded HTTP byte stream as follows: Since the MFCC vectors contain embedded NULL characters, they cannot be 10 transferred in this form to server side system 180 as such using HTTP protocol. Thus the MFCC vectors are first encoded at client-side 150 before transmission in such a way that all the speech data is converted into a stream of bytes without embedded NULL characters in the data. At the very end of the byte stream a single NULL character is introduced to indicate the termination of the stream of bytes to be transferred to the server over the INTERNET 160A using HTTP protocol.

As explained earlier, to conserve latency time between the 20 client and server, a smaller number of bytes Oust the 13 MFCC coefficients) are sent from client side system 150 to server side system 180. This is done automatically for each platform to ensure uniformity, or can be tailored by the particular application environment—i.e., such as where it is determined that it will take less time to compute the delta and acceleration coefficients at the server (26 more calculations), than it would take to encode them at the client, transmit them, and then decode them from the HTTP stream. In general, since server side system 180 is usually better 30 equipped to calculate the MFCC delta and acceleration parameters, this is a preferable choice. Furthermore, there is generally more control over server resources compared to the client's resources, which means that future upgrades, optimizations, etc., can be disseminated and shared by all to 35 make overall system performance more reliable and predictable. So, the present invention can accommodate even the worst-case scenario where the client's machine may be quite thin and may just have enough resources to capture the speech input data and do minimal processing.

Dictionary Preparation & Grammar Files

Referring to FIG. 4A, within code block 605, various options selected by the user (or gleaned from the user's status within a particular application) are received. For 45 instance, in the case of a preferred remote learning system, Course, Chapter and/or Section data items are communicated. In the case of other applications (such as e-commerce) other data options are communicated, such as the Product Class, Product Category, Product Brand, etc. loaded for 50 viewing within his/her browser. These selected options are based on the context experienced by the user during an interactive process, and thus help to limit and define the -i.e. grammars and dictionaries that will be dynamically loaded to speech recognition engine 182 (FIG. 1) for 55 Viterbi decoding during processing of the user speech utterance. For speech recognition to be optimized both grammar and dictionary files are used in a preferred embodiment. A Grammar file supplies the universe of available user queries; i.e., all the possible words that are to be recognized. The Dictionary file provides phonemes (the information of how a word is pronounced—this depends on the specific native language files that are installed—for example, UK English or US English) of each word contained in the grammar file. It is apparent that if all the sentences for a given environment 65 that can be recognized were contained in a single grammar file then recognition accuracy would be deteriorated and the

28

loading time alone for such grammar and dictionary files would impair the speed of the speech recognition process.

To avoid these problems, specific grammars are dynamically loaded or actively configured as the current grammar according to the user's context, i.e., as in the case of a remote learning system, the Course, Chapter and/or Section selected. Thus the grammar and dictionary files are loaded dynamically according to the given Course, Chapter and/or Section as dictated by the user, or as determined automatically by an application program executed by the user.

The second code block 602 implements the initialization of Speech Recognition engine 182 (FIG. 1). The MFCC vectors received from client side system 150 along with the grammar filename and the dictionary file names are introduced to this block to initialize the speech decoder.

As illustrated in FIG. 4A, the initialization process 602 uses the following sub-routines: A routine 602a for loading an SRE library. This then allows the creation of an object identified as External Source with code 602b using the received MFCC vectors. Code 602c allocates memory to hold the recognition objects. Routine 602d then also creates and initializes objects that are required for the recognition such as: Source, Coder, Recognizer and Results Loading of the Dictionary created by code 602e, Hidden Markov Models (HMMs) generated with code 602f; and Loading of the Grammar file generated by routine 602g.

Speech Recognition 603 is the next routine invoked as illustrated in FIG. 4A, and is generally responsible for completing the processing of the user speech signals input on the client side 150, which, as mentioned above, are preferably only partially processed (i.e., only MFCC vectors are computed during the first phase) when they are transmitted across link 160. Using the functions created in External Source by subroutine 602b, this code reads MFCC vectors, one at a time from an External Source 603a, and processes them in block 603b to realize the words in the speech pattern that arc symbolized by the MFCC vectors captured at the client. During this second phase, an additional 13 delta coefficients and an additional 13 acceleration coefficients are computed as part of the recognition process to obtain a total of 39 observation vectors O_t referred to earlier. Then, using a set of previously defined Hidden Markov Models (HMMs), the words corresponding to the user's speech utterance are determined in the manner described earlier. This completes the word "recognition" aspect of the query processing, which results are used further below to complete the query processing operations

It well be appreciated by those skilled in the art that the distributed nature and rapid performance of the word recognition process, by itself, is extremely useful and may be implemented in connection with other environments that do not implicate or require additional query processing operations. For example, some applications may simply use individual recognized words for filling in data items on a computer generated form, and the aforementioned systems and processes can provide a rapid, reliable mechanism for doing so.

Once the user's speech is recognized, the flow of SRE 182 passes to Un-initialize SRE routine 604 where the speech engine is un-initialized as illustrated. In this block all the objects created in the initialization block are deleted by routine 604a, and memory allocated in the initialization block during the initialization phase are removed by routine 604b.

Again, it should be emphasized that the above are merely illustrative of embodiments for implementing the particular

29

routines used on a server side speech recognition system of the present invention. Other variations of the same that achieve the desired functionality and objectives of the present invention will be apparent from the present teachings.

Database Processor 186 Operation—DBProcess

Construction of an SQL Query used as part of the user query processing is illustrated in FIG. 4B, a SELECT SQL statement is preferably constructed using a conventional 10 CONTAINS predicate. Module 950 constructs the SQL query based on this SELECT SQL statement, which query is used for retrieving the best suitable question stored in the database corresponding to the user's articulated query, (designated as Question here). A routine 951 then concatenates a table name with the constructed SELECT statement. Next, the number of words present in each Noun Phrase of Question asked by the user is calculated by routine 952. Then memory is allocated by routine 953 as needed to accommodate all the words present in the NP. Next a word List (identifying all the distinct words present in the NP) is obtained by routine 954. After this, this set of distinct words arc concatenated by routine 955 to the SQL Query separated with a NEAR () keyword. Next, the AND keyword is concatenated to the SQL Query by routine 956 after each NP. Finally memory resources are freed by code 957 so as to allocate memory to store the words received from NP for any next iteration. Thus, at the end of this process, a completed SQL Query corresponding to the user's articulated question is generated.

Connection to SQL Server—As illustrated in FIG. 4C, after the SQL Query is constructed by routine 710, a routine 711 implements a connection to the query database 717 to continue processing of the user query. The connection sequence and the subsequent retrieved record set is implemented using routines 700 which include the following:

- 1. Server and database names are assigned by routine 711A to a DBProcess member variable
- 2. A connection string is established by routine 711B;
- The SQL Server database is connected under control of code 711C
- 4. The SQL Query is received by routine 712A
- 5. The SQL Query is executed by code 712B
- Extract the total number of records retrieved by the query—713
- Allocate the memory to store the total number of paired questions—713
- 8. Store the entire number of paired questions into an 50 array—713

Once the Best Answer ID is received at 716 FIG. 4C, from the NLE 14 (FIG. 5), the code corresponding 716C receives it passes it to code in 716B where the path of the Answer file is determined using the record number. Then the file is 55 opened 716C using the path passed to it and the contents of the file corresponding to the answer is read. Then the answer is compressed by code in 716D and prepared for transmission over the communication channel 160B (FIG. 1).

NLQS Database 188—Table Organization

FIG. 6 illustrates a preferred embodiment of a logical structure of tables used in a typical NLQS database 188 (FIG. 1). When NLQS database 188 is used as part of NLQS query system 100 implemented as a remote learning/training 65 environment, this database will include an organizational multi-level hierarchy that consists typically of a Course 701,

30

which is made of several chapters 702, 703, 704. Each of these chapters can have one or more Sections 705, 706, 707 as shown for Chapter 1. A similar structure can exist for Chapter 2, Chapter 3... Chapter N. Each section has a set of one or more question—answer pairs 708 stored in tables described in more detail below. While this is an appropriate and preferable arrangement for a training/learning application, it is apparent that other implementations would be possible and perhaps more suitable for other applications such as e-commerce, e-support, INTERNET browsing, etc., depending on overall system parameters.

It can be seen that the NLQS database 188 organization is intricately linked to the switched grammar architecture described earlier. In other words, the context (or environment) experienced by the user can be determined at any moment in time based at the selection made at the section level, so that only a limited subset of questionanswer pairs 708 for example are appropriate for section 705. This in turn means that only a particular appropriate grammar for such question-answer pairs may be switched in for handling user queries while the user is experiencing such context. In a similar fashion, an e-commerce application for an INTERNET based business may consist of a hierarchy that includes a first level "home" page 701 identifying user selectable options (product types, services, contact information, etc.), a second level may include one or more "product types" pages 702, 703, 704, a third page may include particular product models 705, 706, 707, etc., and with appropriate question-answer pairs 708 and grammars customized for handling queries for such product models. Again, the particular implementation will vary from application to application, depending on the needs and desires of such business, and a typical amount of routine optimization will be necessary for each such application.

Table Organization

In a preferred embodiment, an independent database is used for each Course. Each database in turn can include three types of tables as follows: a Master Table as illustrated in FIG. 7A, at least one Chapter Table as illustrated in FIG. 7B and at least one Section Table as illustrated in FIG. 7C.

As illustrated in FIG. 7A, a preferred embodiment of a Master Table has six columns—Field Name 701A, Data Type 702A, Size 703A, Null 704A, Primary Key 705A and Indexed 706A. These parameters are well-known in the art of database design and structure. The Master Table has only two fields—Chapter Name 707A and Section Name 708A. Both ChapterName and Section Name are commonly indexed.

A preferred embodiment of a Chapter Table is illustrated in FIG. 7B. As with the Master Table, the Chapter Table has six (6) columns—Field Name 720, Data Type 721, Size 722, Null 723, Primary Key 724 and Indexed 725. There are nine (9) rows of data however, in this case,—Chapter_ID 726, Answer_ID 727, Section Name 728, Answer_Tide 729, PairedQuestion 730, AnswerPath 731, Creator 732, Date of Creation 733 and Date of Modification 734.

An explanation of the Chapter Table fields is provided in FIG. 7C. Each of the eight (8) Fields 720 has a description 735 and stores data corresponding to:

AnswerID 727—an integer that is automatically incremented for each answer given for user convenience

Section_Name 728—the name of the section to which the particular record belongs. This field along with the AnswerlD is used as the primary key

Answer_Title **729**—A short description of the title of the answer to the user query

10

31

- PairedQuestion 730—Contains one or more combinations of questions for the related answers whose path is stored in the next column AnswerPath
- AnswerPath 731—contains the path of a file, which contains the answer to the related questions stored in 5 the previous column; in the case of a pure question/ answer application, this file is a text file, but, as mentioned above, could be a multi-media file of any kind transportable over the data link 160
- Creator 732—Name of Content Creator
- Date_of_Creation 733—Date on which content was created
- Date of Modification 734—Date on which content was changed or modified

A preferred embodiment of a Section Table is illustrated 15 in FIG. 7D. The Section Table has six (6) columns—Field Name 740, Data Type 741, Size 742, Null 743, Primary Key 744 and Indexed 745. There are seven (7) rows of data-Answer_ID 746, Answer_Title 747, PairedQuestion 748, AnswerPath 749, Creator 750, Date of Creation 751 and 20 Date of Modification 752. These names correspond to the same fields, columns already described above for the Master Table and Chapter Table.

Again, this is a preferred approach for the specific type of learning/training application described herein. Since the 25 number of potential applications for the present invention is quite large, and each application can be customized, it is expected that other applications (including other learning/ training applications) will require and/or be better accommodated by another table, column, and field structure/ 30 hierarchy.

Search Service and Search Engine—A query text search service is performed by an SQL Search System 1000 shown in FIG. 10. This system provides querying support to process full-text searches. This is where full-text indexes reside.

In general, SQL Search System determines which entries in a database index meet selection criteria specified by a particular text query that is constructed in accordance with an articulated user speech utterance. The Index Engine **1011**B is the entity that populates the Full-Text Index tables 40 with indexes which correspond to the indexable units of text for the stored questions and corresponding answers. It scans through character strings, determines word boundaries, removes all noise words and then populates the full-text index with the remaining words. For each entry in the full 45 text database that meets the selection criteria, a unique key column value and a ranking value are returned as well. Catalog set 1013 is a file-system directory that is accessible only by an Administrator and Search Service 1010. Full-text indexes 1014 are organized into full-text catalogs, which are 50 referenced by easy to handle names. Typically, full-text index data for an entire database is placed into a single full-text catalog.

The schema for the full-text database as described (FIG. 7, FIG. 7A, FIG. 7B, FIG. 7C, FIG. 7D) is stored in the 55 tables 1006 shown in FIG. 10. Take for example, the tables required to describe the structure the stored question/answer pairs required for a particular course. For each table— Course Table, Chapter Table, Section Table, there are fields—column information that define each parameters that 60 make up the logical structure of the table. This information is stored in User and System tables 1006. The key values corresponding to those tables are stored as Full-Text catalogs 1013. So when processing a full-text search, the search engine returns to the SQL Server the key values of the rows 65 that match the search criteria. The relational engine then uses this information to respond to the query.

32

As illustrated in FIG. 10, a Full-Text Query Process is implemented as follows:

- 1. A query 1001 that uses a SQL full-text construct generated by DB processor 186 is submitted to SQL Relational Engine 1002.
- 2. Queries containing either a CONTAINS or FREETEXT predicate are rewritten by routine 1003 so that a responsive rowset returned later from Full-Text Provider 1007 will be automatically joined to the table that the predicate is acting upon. This rewrite is a mechanism used to ensure that these predicates are a seamless extension to a traditional SQL Server. After the compiled query is internally rewritten and checked for correctness in item 1003, the query is passed to RUN TIME module 1004. The function of module 1004 is to convert the rewritten SQL construct to a validated run-time process before it is sent to the Full-Text Provider, 1007.
- 3. After this, Full-Text Provider 1007 is invoked, passing the following information for the query:
- a. A ft_search_condition parameter (this is a logical flag indicating a full text search condition)
- b. A name of a full-text catalog where a full-text index of a table resides
- c. A locale ID to be used for language (for example, word breaking)
- d. Identities of a database, table, and column to be used in the query
- e. If the query is comprised of more than one full-text construct; when this is the case Full-text provider 1007 is invoked separately for each construct.
- 4. SQL Relational Engine 1002 does not examine the contents of ft_search_condition. Instead, this information is passed along to Full-text provider 1007, which verifies the validity of the query and then creates an appropriate internal representation of the full-text search condition.
- 5. The query request/command 1008 is then passed to Querying Support 1011A.
- 6. Querying Support 1012 returns a rowset 1009 from Full-Text Catalog 1013 that contains unique key column values for any rows that match the full-text search criteria. A rank value also is returned for each row.
- 7. The rowset of key column values 1009 is passed to SQL Relational Engine 1002. If processing of the query implicates either a CONTAINSTABLE() or FREETEXTTABLE() function, RANK values are returned; otherwise, any rank value is filtered out.
- 8. The rowset values 1009 are plugged into the initial query with values obtained from relational database 1006, and a result set 1015 is then returned for further processing to yield a response to the user.

At this stage of the query recognition process, the speech utterance by the user has already been rapidly converted into a carefully crafted text query, and this text query has been initially processed so that an initial matching set of results can be further evaluated for a final determination of the appropriate matching question/answer pair. The underlying principle that makes this possible is the presence of a full-text unique key column for each table that is registered for full-text searches. Thus when processing a full-text search, SQL Search Service 1010 returns to SQL server 1002 the key values of the rows that match the database. In maintaining these full-text databases 1013 and full text indexes 1014, the present invention has the unique characteristic that the full-text indices 1014 are not updated instantly when the full-text registered columns are updated.

33

This operation is eliminated, again, to reduce recognition latency, increase response speed, etc. Thus, as compared to other database architectures, this updating of the full-text index tables, which would otherwise take a significant time, is instead done asynchronously at a more convenient time. 5

Interface Between NLE 190 and DB Processor 188

The result set 1015 of candidate questions corresponding to the user query utterance are presented to NLE 190 for further processing as shown in FIG. 4D to determine a "best" matching question/answer pair. An NLE/ DBProcessor interface module coordinates the handling of user queries, analysis of noun-phrases (NPs) of retrieved questions sets from the SQL query based on the user query, comparing the retrieved question NPs with the user query NP, etc. between NLE 190 and DB Processor 188. So, this part of the server side code contains functions, which interface processes resident in both NLE block 190 and DB Processor block 188. The functions are illustrated in FIG. 4D; As seen here, code routine 880 implements functions to extract the Noun Phrase (NP) list from the user's question. This part of the code interacts with NLE 190 and gets the list of Noun Phrases in a sentence articulated by the user. Similarly, Routine 813 retrieves an NP list from the list of corresponding candidate/paired questions 1015 and stores these questions into an (ranked by NP value) array. Thus, at this point, NP data has been generated for the user query, as well as for the candidate questions 1015. As an example of determining the noun phrases of a sentence such as: "What issues have guided the President in considering the impact of foreign trade policy on American businesses?" NLE 190 would return the following as noun phrases: President, issues, impact of foreign trade policy, American businesses, impact, impact of foreign trade, foreign trade, foreign trade policy, trade, trade policy, policy, businesses. The methodology used by NLE 190 will thus be apparent to those skilled in the art from this set of noun phrases and noun sub-phrases generated in response to the example query.

implemented. This part of the code gets a best answer ID corresponding to the user's query. To do this, routines 813A, 813B first find out the number of Noun phrases for each entry in the retrieved set 1015 that match with the Noun phrases in the user's query. Then routine **815***a* selects a final result record from the candidate retrieved set 1015 that contains the maximum number of matching Noun phrases.

Conventionally, nouns are commonly thought of as "naming" words, and specifically as the names of "people, places, or things". Nouns such as John, London, and computer 50 and a length. The first phase of tokenization is segmentation, certainly fit this description, but the types of words classified by the present invention as nouns is much broader than this. Nouns can also denote abstract and intangible concepts such as birth, happiness, evolution, technology, management, imagination, revenge, politics, hope, cookery, sport, and literacy. Because of the enormous diversity of nouns compared to other parts of speech, the Applicant has found that it is much more relevant to consider the noun phrase as a key linguistic metric. So, the great variety of items classified as nouns by the present invention helps to discriminate and identify individual speech utterances much easier and faster than prior techniques disclosed in the art.

Following this same thought, the present invention also adopts and implements another linguistic entity—the word phrase—to facilitate speech query recognition. The basic 65 it does not have parts of speech information. Analyzer 806B structure of a word phrase—whether it be a noun phrase, verb phrase, adjective phrase—is three parts—[pre-Head

34

string],[Head] and [post-Head string]. For example, in the minimal noun phrase—"the children," "children" is classified as the Head of the noun phrase. In summary, because of the diversity and frequency of noun phrases, the choice of noun phrase as the metric by which stored answer is linguistically chosen, has a solid justification in applying this technique to the English natural language as well as other natural languages. So, in sum, the total noun phrases in a speech utterance taken together operate extremely well as unique type of speech query fingerprint.

The ID corresponding to the best answer corresponding to the selected final result record question is then generated by routine 815 which then returns it to DB Process shown in FIG.4C. As seen there, a Best Answer ID I is received by routine 716A, and used by a routine 716B to retrieve an answer file path. Routine 716C then opens and reads the answer file, and communicates the substance of the same to routine 716D. The latter then compresses the answer file data, and sends it over data link 160 to client side system 150 for processing as noted earlier (i.e., to be rendered into audible feedback, visual text/graphics, etc.). Again, in the context of a learning/instructional application, the answer file may consist solely of a single text phrase, but in other applications the substance and format will be tailored to a specific question in an appropriate fashion. For instance, an "answer" may consist of a list of multiple entries corresponding to a list of responsive category items (i.e., a list of books to a particular author) etc. Other variations will be apparent depending on the particular environment.

Natural Language Engine 190

Again referring to FIG. 4D, the general structure of NL engine 190 is depicted. This engine implements the word analysis or morphological analysis of words that make up the user's query, as well as phrase analysis of phrases extracted from the query.

As illustrated in FIG. 9, the functions used in a morphological analysis include tokenizers 802A, stemmers 804A and morphological analyzers 806A. The functions that com-Next, a function identified as Get Best Answer ID 815 is 40 prise the phrase analysis include tokenizers, taggers and groupers, and their relationship is shown in FIG.8.

> Tokenizer 802A is a software module that functions to break up text of an input sentence 801A into a list of tokens 803A. In performing this function, tokenizer 802A goes through input text 801A and treats it as a series of tokens or useful meaningful units that are typically larger than individual characters, but smaller than phrases and sentences. These tokens 803A can include words, separable parts of word and punctuation. Each token 803A is given an offset which extracts the individual tokens from the input text and keeps track of the offset where each token originated from in the input text. Next, categories are associated with each token, based on its shape. The process of tokenization is well-known in the art, so it can be performed by any convenient application suitable for the present invention.

> Following tokenization, a stemmer process 804A is executed, which can include two separate formsinflectional and derivational, for analyzing the tokens to determine their respective stems 805A. An inflectional stemmer recognizes affixes and returns the word which is the stem. A derivational stemmer on the other hand recognizes derivational affixes and returns the root word or words. While stemmer 804A associates an input word with its stem, takes a word independent of context, and returns a set of possible parts of speech 806A.

35

As illustrated in FIG. 8, phrase analysis 800 is the next step that is performed after tokenization. A tokenizer 802 generates tokens from input text 801. Tokens 803 are assigned to parts of a speech tag by a tagger routine 804, and a grouper routine 806 recognizes groups of words as phrases of a certain syntactic type. These syntactic types include for example the noun phrases mentioned earlier, but could include other types if desired such as verb phrases and adjective phrases. Specifically, tagger 804 is a parts-ofspeech disambiguator, which analyzes words in context. It 10 has a built-in morphological analyzer (not shown) that allows it to identify all possible parts of speech for each token. The output of tagger 804 is a string with each token tagged with a parts-of-speech label 805. The final step in the linguistic process 800 is the grouping of words to form 15 phrases 807. This function is performed by the grouper 806, and is very dependent, of course, on the performance and output of tagger component 804.

Accordingly, at the end of linguistic processing 800, a list of noun phrases (NP) 807 is generated in accordance with 20 the user's query utterance. This set of NPs generated by NLE 190 helps significantly to refine the search for the best answer, so that a single-best answer can be later provided for the user's question.

The particular components of NLE 190 are shown in FIG. 25 4D, and include several components. Each of these components implement the several different functions required in NLE 190 as now explained.

Initialize Grouper Resources Object and the Library 900—this routine initializes the structure variables required to create grouper resource object and library. Specifically, it initializes a particular natural language used by NLE 190 to create a Noun Phrase, for example the English natural language is initialized for a system that serves the English language market. In turn, it also creates the objects (routines) required for Tokenizer, Tagger and Grouper (discussed above) with routines 900A, 900B, 900C and 900D respectively, and initializes these objects with appropriate values. It also allocates memory to store all the recognized Noun Phrases for the retrieved question pairs.

Tokenizing of the words from the given text (from the query or the paired questions) is performed with routine 909B—here all the words are tokenized with the help of a local dictionary used by NLE 190 resources. The resultant tokenized words are passed to a Tagger routine 909C. At routine 909C, tagging of all the tokens is done and the output is passed to a Grouper routine 909D.

The Grouping of all tagged token to form NP list is the tagged token words and outputs the Noun Phrases.

Un-initializing of the grouper resources object and freeing of the resources, is performed by routines 909EA, 909EB and 909EC. These include Token Resources, Tagger Resources and Grouper Resources respectively. After 55 initialization, the resources are freed. The memory that was used to store all Noun Phrases are also de-allocated.

Additional Embodiments

In a e-commerce embodiment of the present invention as 60 illustrated in FIG. 13, a web page 1300 contains typical visible links such as Books 1310, Music 1320 so that on clicking the appropriate link the customer is taken to those pages. The web page may be implemented using HTML, a Java applet, or similar coding techniques which interact with the user's browser. For example, if customer wants to buy an album C by Artist Albert, he traverses several web pages as

36

follows: he first clicks on Music (FIG. 13, 1360), which brings up page 1400 where he/she then clicks on Records (FIG. 14, 1450). Alternatively, he/she could select CDs 1460, Videos 1470, or other categories of books 1410, music 1420 or help 1430. As illustrated in FIG. 15, this brings up another web page 1500 with links for Records 1550, with sub-categories—Artist 1560, Song 1570, Tide 1580, Genre 1590. The customer must then click on Artist 1560 to select the artist of choice. This displays another web page 1600 as illustrated in FIG. 16. On this page the various artists 1650 are listed as illustrated—Albert 1650, Brooks 1660, Charlie 1670, Whyte 1690 are listed under the category Artists 1650. The customer must now click on Albert 1660 to view the albums available for Albert. When this is done, another web page is displayed as shown in FIG. 17. Again this web page 1700 displays a similar look and feel, but with the albums available 1760, 1770, 1780 listed under the heading Titles 1750. The customer can also read additional information 1790 for each album. This album information is similar to the liner notes of a shrink-wrapped album purchased at a retail store. One Album A is identified, the customer must click on the Album A 1760. This typically brings up another text box with the information about its availability, price, shipping and handling charges etc.

When web page 1300 is provided with functionality of a NLQS of the type described above, the web page interacts with the client side and server side speech recognition modules described above. In this case, the user initiates an inquiry by simply clicking on a button designated Contact Me for Help 1480 (this can be a link button on the screen, or a key on the keyboard for example) and is then told by character 1440 about how to elicit the information required. If the user wants Album A by artist Albert, the user could articulate "Is Album A by Brooks available?" in much the same way they would ask the question of a human clerk at a brick and mortar facility. Because of the rapid recognition performance of the present invention, the user's query would be answered in real-time by character 1440 speaking out the answer in the user's native language. If desired, a 40 readable word balloon 1490 could also be displayed to see the character's answer and so that save/print options can also be implemented. Similar appropriate question/answer pairs for each page of the website can be constructed in accordance with the present teachings, so that the customer is provided with an environment that emulates a normal conversational human-like question and answer dialog for all aspects of the web site. Character 1440 can be adjusted and tailored according to the particular commercial application, or by the user's own preferences, etc. to have a particular implemented by routine 909D so that the Grouper groups all 50 voice style (man, woman, young, old, etc.) to enhance the customer's experience.

In a similar fashion, an articulated user query might be received as part of a conventional search engine query, to locate information of interest on the INTERNET in a similar manner as done with conventional text queries. If a reasonably close question/answer pair is not available at the server side (for instance, if it does not reach a certain confidence level as an appropriate match to the user's question) the user could be presented with the option of increasing the scope so that the query would then be presented simultaneously to one or more different NLEs across a number of servers, to improve the likelihood of finding an appropriate matching question/answer pair. Furthermore, if desired, more than one "match" could be found, in the same fashion that conventional search engines can return a:number of potential "hits" corresponding to the user's query. For some such queries, of course, it is likely that real-time performance will not be

37

possible (because of the disseminated and distributed processing) but the advantage presented by extensive supplemental question/answer database systems may be desirable for some users.

It is apparent as well that the NLQS of the present 5 invention is very natural and saves much time for the user and the e-commerce operator as well. In an e-support embodiment, the customer can retrieve information quickly and efficiently, and without need for a live customer agent. For example, at a consumer computer system vendor related support site, a simple diagnostic page might be presented for the user, along with a visible support character to assist him/her. The user could then select items from a "symptoms" page (i.e., a "monitor" problem, a "keyboard" problem, a "printer" problem, etc.) simply by articulating 15 such symptoms in response to prompting from the support character. Thereafter, the system will direct the user on a real-time basis to more specific sub-menus, potential solutions, etc. for the particular recognized complaint. The use of a programmable character thus allows the web site to be scaled to accommodate a large number of hits or customers without any corresponding need to increase the number of human resources and its attendant training issues.

As an additional embodiment, the searching for information on a particular web site may be accelerated with the use 25 of the NLQS of the present invention. Additionally, a significant benefit is that the information is provided in a user-friendly manner through the natural interface of speech. The majority of web sites presently employ lists of frequently asked questions which the user typically wades item 30 by item in order to obtain an answer to a question or issue. For example, as displayed in FIG. 13, the customer clicks on Help 1330 to initiate the interface with a set of lists. Other options include computer related items at 1370 and frequently asked questions (FAQ) at 1380.

As illustrated in FIG. 18, a web site plan for typical web page is displayed. This illustrates the number of pages that have to be traversed in order to reach the list of Frequently-Asked Questions. Once at this page, the user has to scroll query. This process is typically a laborious task and may or may not yield the information that answers the user's query. The present art for displaying this information is illustrated in FIG. 18. This figure identifies how the information on a typical web site is organized: the Help link (FIG. 13, 1330) typically shown on the home page of the web page is illustrated shown on FIG. 18 as 1800. Again referring to FIG. 18, each sub-category of information is listed on a separate page. For example, 1810 lists sub-topics such as 'First Time Visitors', 'Search Tips', 'Ordering', 'Shipping', 50 'Your Account' etc. Other pages deal with 'Account information' 1860, 'Rates and Policies' 1850 etc. Down another level, there are pages that deal exclusively with a sub-sub topics on a specific page such as 'First Time Visitors' 1960, 'Frequently Asked Questions' 1950, 'Safe Shopping Guar- 55 antee' 1940, etc. So if a customer has a query that is best answered by going to the Frequently Asked Questions link, he or she has to traverse three levels of busy and cluttered screen pages to get to the Frequently Asked Questions page 1950. Typically, there are many lists of questions 1980 that have to be manually scrolled through. While scrolling visually, the customer then has to visually and mentally match his or her question with each listed question. If a possible match is sighted, then that question is clicked and the answer then appears in text form which then is read.

In contrast, the process of obtaining an answer to a question using a web page enabled with the present NLQS 38

can be achieved much less laboriously and efficiently. The user would articulate the word "Help" (FIG. 13, 1330). This would immediately cause a character (FIG. 13, 1340) to appear with the friendly response "May I be of assistance. Please state your question?". Once the customer states the question, the character would then perform an animation or reply "Thank you, I will be back with the answer soon". After a short period time (preferably not exceeding 5-7 seconds) the character would then speak out the answer to the user's question. As illustrated in FIG. 18 the answer would be the answer 1990 returned to the user in the form of speech is the answer that is paired with the question 1950. For example, the answer 1990: "We accept Visa, MasterCard and Discover credit cards", would be the response to the query 2000 "What forms of payments do you accept?"

Another embodiment of the invention is illustrated in FIG. 12. This web page illustrates a typical website that employs NLQS in a web-based learning environment. As illustrated in FIG. 12, the web page in browser 1200, is divided into two or more frames. A character 1210 in the likeness of an instructor is available on the screen and appears when the student initiates the query mode either by speaking the word "Help" into a microphone (FIG. 2, 215) or by clicking on the link 'Click to Speak' (FIG. 12, 1280). Character 1210 would then prompt the student to select a course 1220 from the drop down list 1230. If the user selects the course 'CPlusPlus', the character would then confirm verbally that the course "CPlusPlus" was selected. The character would then direct the student to make the next selection from the drop-down list 1250 that contains the selections for the chapters 1240 from which questions are available. Again, after the student makes the selection, the character 1210 confirms the selection by speaking. Next character 1210 prompts the student to select 'Section' 1260 of the chapter from which questions are available from the drop down list 1270. Again, after the student makes the selection, character 1210 confirms the selection by articulating the 'Section' 1260 chosen. As a prompt to the student, a list of possible questions appear in the list box 1291. In and manually identify the question that matches his/her 40 addition, tips 1290 for using the system are displayed. Once the selections are all made, the student is prompted by the character to ask the question as follows: "Please ask your query now". The student then speaks his query and after a short period of time, the character responds with the answer preceded by the question as follows: "The answer to your question . . . is as follows: . . . ". This procedure allows the student to quickly retrieve answers to questions about any section of the course and replaces the tedium of consulting books, and references or indices. In short, it is can serve a number of uses from being a virtual teacher answering questions on-the-fly or a flash card substitute.

From preliminary data available to the inventors, it is estimate that the system can easily accommodate 100-250 question/answer pairs while still achieving a real-time feel and appearance to the user (i.e., less than 10 seconds of latency, not counting transmission) using the above described structures and methods. It is expected, of course, that these figures will improve as additional processing speed becomes available, and routine optimizations are employed to the various components noted for each particular environment.

Again, the above are merely illustrative of the many possible applications of the present invention, and it is expected that many more web-based enterprises, as well as other consumer applications (such as intelligent, interactive toys) can utilize the present teachings. Although the present invention has been described in terms of a preferred

39

embodiment, it will be apparent to those skilled in the art that many alterations and modifications may be made to such embodiments without departing from the teachings of the present invention. It will also be apparent to those skilled in the art that many aspects of the present discussion have been simplified to give appropriate weight and focus to the more germane aspects of the present invention. The microcode and software routines executed to effectuate the inventive methods may be embodied in various forms, including in a permanent magnetic media, a non-volatile ROM, a CD-ROM, or any other suitable machine-readable format. Accordingly, it is intended that the all such alterations and modifications be included within the scope and spirit of the invention as defined by the following claims.

What is claimed is:

- 1. An interactive learning system adapted for responding to speech-based queries concerning topics addressed by such interactive learning system, the system comprising:
 - a query file for storing a plurality of topic query entries, each topic query entry including a query relating to one or more of the topics covered by the speech-based interactive learning system; and
 - an answer file for storing a plurality of topic answer entries, each topic answer entry including an answer to one or more of said plurality of topic query entries, such that each topic query entry has at least one associated topic answer entry; and
 - a speech recognition system for generating recognized speech utterance data from partially processed speech data associated with a speech-based query concerning one of said topics, said partially processed speech data being received from a remote speech capturing system; and
 - said speech recognition system further cooperating with a natural language engine, which processes said recognized speech utterance data using a morphological analysis and a phrase analysis to form recognized speech sentence data corresponding to said speech-based query;
 - a query formulation system for converting said recognized speech sentence data into a search query suitable for identifying a topic query entry corresponding to said speech-based query, and for locating at least one topic answer entry best matching said speech-based query.

 45
- 2. The system of claim 1, wherein said remote speech capturing system is located at a client site, and said speech recognition system is distributed across said client site and a separate server site.
- 3. The system of claim 1, wherein said speech recognition 50 system is comprised of a first portion at a client based computing system for performing first signal processing operations on a speech input signal to create said partially processed speech data, and a second portion at a server based computing system for performing a second signal 55 processing operation for completing processing of said partially processed speech data.
- 4. The system of claim 1, wherein said query formulation system uses context parameters for recognizing said speech-based query.
- 5. The system of claim 4, wherein said context parameters include any one or more of the following: a course, chapter, and/or section of said lesson information selected by said user during the interactive session.
- 6. The system of claim 4, wherein said context parameters 65 include any one or more visible text words or objects presented to said user during the interactive session.

40

- 7. The system of claim 4, wherein said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.
- 8. A speech based interactive learning system comprising: an instructional file containing instructional materials arranged in a hierarchical structure that includes at least a first level of instruction data and a second level of instruction data; and
- wherein users of such system can navigate said hierarchical structure and formulate a user query concerning instructional materials located at least at said second level of instruction data;
- further wherein said user query can be correlated with a corresponding instructional material question taken from a list of predefined questions, said set of predefined questions being paired with a corresponding set of responsive answers;
- a speech recognition engine for generating recognized words from a user query;
- a natural language engine for parsing said words contained in said user query to generate recognized speech sentence data;
- a query formulation engine for converting said recognized speech sentence data into a search query suitable for identifying a corresponding instructional material question for said user query, and for locating a corresponding responsive answer for said corresponding instructional material question.
- 9. A speech based interactive learning system comprising: an instructional file containing instructional materials arranged in a hierarchical structure that includes at least a first level of instruction data and a second level of instruction data; and
- wherein users of such system can navigate said hierarchical structure and formulate a user query concerning instructional materials located at least at said second level of instruction data;
- further wherein said user query can be correlated with a corresponding instructional material question taken from a list of predefined questions, said set of predefined questions being paired with a corresponding set of responsive answers;
- a speech recognition engine for generating recognized speech data from a user query;
- a query formulation engine for converting said recognized speech data into a search query suitable for identifying a corresponding instructional material question for said user query, and for locating a corresponding responsive answer for said corresponding instructional material question;
- wherein said search query is formulated using a natural language engine parsing words contained in said user query.
- 10. The system of claim 9, wherein said natural language engine compares said user query with one or more of said list of predefined questions to determine said corresponding instructional material question.
- 11. The system of claim 8, where a set of potential questions corresponding to said user query is derived from said list of predefined questions by partially recognizing said user query, and said corresponding instructional material question is derived by fully recognizing said user query from said set of potential questions.
- 12. The system of claim 9, wherein said first level of instruction data is associated with one or more lesson

41

chapters for a particular course, and said second level of instruction data is associated with one or more sections, said one or more sections being linked to said one or more chapters.

- 13. The system of claim 8, further including an animated 5 visible agent for assisting said user to navigate said instructional materials, and for articulating said corresponding responsive answer in audible form to said user.
- 14. The system of claim 9, further including an animated visible agent for assisting said user to navigate said instruc- 10 query system, including the steps of: tional materials, and for articulating said corresponding responsive answer to said user.
- 15. A speech based system for assisting a user in connection with an interactive lesson tutorial having question and answer capability, the system comprising:
 - a lesson file containing instructional materials arranged to include a list of predefined questions and a corresponding list of predefined answers for said lesson; and
 - a speech recognition engine for generating recognized speech word data from a user query pertaining to said 20 lesson; and
 - a natural language engine for generating recognized speech sentence data from said speech word data; and
 - a query recognition engine for locating a corresponding 25 predefined question for said user query using said recognized speech sentence data; and
 - a conversion engine for converting a corresponding predefined answer for said corresponding predefined question into a form perceptible to the user;
 - said query recognition system being adapted to process said user query, even before said user has completed articulating said user query, to identify said corresponding predefined answer and thus emulate a human response time in response to a user query, so that the 35 user perceives interaction with such system in essentially the same way that would be experienced from interacting with a real human.
- 16. A speech based system for assisting a user in connection with an interactive lesson tutorial having question and $\,^{40}$ answer capability, the system comprising:
 - a lesson file containing instructional materials arranged to include a list of predefined questions and a corresponding list of predefined answers for said lesson; and
 - a speech recognition engine for generating recognized 45 speech data from a user query pertaining to said lesson;
 - a query recognition engine for locating a corresponding predefined question for said user query using said 50 recognized speech data; and
 - a conversion engine for converting a corresponding predefined answer for said corresponding predefined question into a form perceptible to the user;
 - said query recognition system being adapted to emulate a 55 human response time in response to a user query, so that the user perceives interaction with such system in essentially the same way that would be experienced from interacting with a real human.
- 17. The system of claim 16, wherein said system responds 60 to a user query in less than approximately 10 seconds for a list of predefined questions having at least 100 entries, not accounting for data transmission latencies between the user and the system.
- **18**. The system of claim **16**, wherein said system further 65 includes a visible interactive agent that receives and responds to said user query.

42

- 19. The system of claim 16, wherein said interactive agent is configured with an appearance and mannerism that emulates a corresponding human appearance and mannerism appropriate for said lesson tutorial.
- 20. The system of claim 15, wherein said query recognition engine is distributed across multiple computing systems so that said more than one lesson file is consulted for said
- 21. A method of implementing a speech-based interactive
 - (a) storing a plurality of topic query entries, each topic query entry including a query relating to one or more of topics covered by the speech-based interactive learning system: and
 - (b) storing a plurality of topic answer entries, each topic answer entry including an answer to one or more of said plurality of topic query entries, such that each topic query entry has at least one associated topic answer entry; and
 - (c) generating recognized speech utterance data associated with a speech-based query concerning one of said topics, such that said recognized speech utterance data is generated by partial recognition processing of said speech-based query by a first signal processing routine executing at a first computing device, and then completing recognition of said speech-based query through processing performed by a second signal processing routine executing at a second computing device; and
 - (d) converting said recognized speech utterance data with a natural language process into recognized speech sentence data, said recognized speech data being used by a search query suitable for identifying a topic query entry corresponding to said speech-based query; and
 - (e) locating at least one topic answer entry best matching said speech-based query;
 - wherein step (e) is initiated before said natural language engine has converted said recognized speech utterance data into recognized speech sentence data.
- 22. A method of implementing a speech-based interactive query system, including the steps of:
 - (a) storing a plurality of topic query entries, each topic query entry including a query relating to one or more of topics covered by the speech-based interactive learning
 - (b) storing a plurality of topic answer entries, each topic answer entry including an answer to one or more of said plurality of topic query entries, such that each topic query entry has at least one associated topic answer entry; and
 - (c) generating recognized speech data associated with a speech-based query concerning one of said topics, such that said recognized speech data is generated by partial recognition processing of said speech-based query by a first signal processing routine executing at a first computing device, and then completing recognition of said speech-based query through processing performed by a second signal processing routine executing at a second computing device; and
 - (d) converting said recognized speech data into a search query suitable for identifying a topic query entry corresponding to said speech-based query; and
 - (e) locating at least one topic answer entry best matching said speech-based query
- 23. The method of claim 21, wherein during step (d) context parameters are used for formulating said search

Filed 09/08/2008

43

query, and said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.

- 24. The method of claim 1, wherein during step (d) context parameters are used for formulating said search query, and said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.
- **25**. A method of presenting an interactive lesson to a user comprising the steps of:
 - (a) arranging instructional materials in a hierarchical structure that includes at least a first level of instruction data and a second level of instruction data; and
 - (b) presenting the instructional materials to a user so that the user can navigate said hierarchical structure and 15 formulate a user query concerning instructional materials located at least at said second level of instruction data;
 - (c) defining a list of predefined questions for at least said second level of instruction data, said set of predefined 20 questions being paired with a corresponding set of responsive answers;
 - (d) generating recognized speech utterance data from a user query for such interactive lesson;
 - (e) converting said recognized speech utterance data into 25 a search query suitable for identifying a corresponding instructional material question for said user query, said search query being formulated with assistance from a natural language engine parsing words in said recognized speech utterance data; and 30
 - (f) identifying a corresponding responsive answer for said corresponding instructional material question;
 - wherein a set of potential questions corresponding to said user query is derived by partially processing said speech utterance data during step (e), and then said corresponding instructional material question is determined by fully processing said speech utterance data.
- 26. The method of claim 25, wherein said response undergoes a text to speech process so that said topic answer entry is expressed in audible form to a user.
- 27. A method of presenting an interactive lesson to a user comprising the steps of:
 - (a) arranging instructional materials in a hierarchical structure that includes at least a first level of instruction data and a second level of instruction data; and
 - (b) presenting the instructional materials to a user so that the user can navigate said hierarchical structure and formulate a user query concerning instructional materials located at least at said second level of instruction data;
 - (c) defining a list of predefined questions for at least said second level of instruction data, said set of predefined questions being paired with a corresponding set of responsive answers;
 - (d) generating recognized speech data from a user query for such interactive lesson;
 - (e) converting said recognized speech data into a search query suitable for identifying a corresponding instructional material question for said user query, said search query being formulated with assistance from a natural language engine parsing words in said recognized speech data; and
 - (f) identifying a corresponding responsive answer for said corresponding instructional material question.
- 28. The method of claim 27, wherein during step (f) said natural language engine operates on said recognized speech

44

data as well as on one or more of said list of predefined questions to determine said corresponding instructional material question.

- 29. The method of claim 27, wherein said first level of instruction data is associated with one or more lesson chapters for a particular course, and said second level of instruction data is associated with one or more sections, said one or more sections being linked to said one or more chapters.
- **30**. A method of operating a speech-based lesson tutorial system having question and answer capability, the method comprising the steps of:
 - (a) configuring a list of retrievable predefined questions and a corresponding list of predefined answers for said lesson; and
 - (b) generating recognized speech data from a user query pertaining to said lesson; and
 - wherein said recognized speech data is generated by a combination of both speech utterance recognition and natural language processing;
 - (c) locating a corresponding predefined question for said user query using said recognized speech data; and
 - (d) converting a corresponding predefined answer for said corresponding predefined question into a form perceptible to the user; and
 - wherein at least step (b) is performed at least in part while a user is articulating said query so as to emulate a human response time in response to a user query, so that the user perceives interaction with such system in essentially the same way that would be experienced from interacting with a real human.
- 31. The method of claim 27, further including a step (g): displaying an animated visible agent for assisting said user to navigate said instructional materials, and for articulating said corresponding responsive answer to said user.
- 32. The method of claim 27, wherein during step (e) context parameters are used for recognizing said speech-based query, and said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.
- **33.** A method of operating a speech-based lesson tutorial system having question and answer capability, the method comprising the steps of:
- (a) configuring a list of retrievable predefined questions and a corresponding list of predefined answers for said lesson; and
- (b) generating recognized speech data from a user query pertaining to said lesson; and
- (c) locating a corresponding predefined question for said user query using said recognized speech data; and
- (d) converting a corresponding predefined answer for said corresponding predefined question into a form perceptible to the user; and
 - wherein steps (b) to (d) are performed so as to emulate a human response time in response to a user query, so that the user perceives interaction with such system in essentially the same way that would be experienced from interacting with a real human.
- **34**. The method of claim **33**, wherein step (d) occurs in response to a user query in less than approximately 10 seconds for a list of predefined questions having at least 100 entries, not accounting for data transmission latencies between the user and the tutorial system.
- 35. The method of claim 30, wherein step (c) occurs across multiple computing systems so that said more than one lesson file is consulted for said user query.

45

- **36**. The method of claim **35**, wherein said interactive agent is configured with an appearance and mannerism that emulates a corresponding human appearance and mannerism appropriate for said lesson tutorial.
- 37. The method of claim 33, wherein during step (c) context parameters are used for recognizing said speech-based query, and said context parameters are used for

46

dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.

38. The method of claim 33, wherein step (c) occurs across multiple computing system so that said more than one lesson file is consulted for said user query.

* * * *

KENT DECLARATION EXHIBIT 3

US007050977B1

(12) United States Patent Bennett

(54) SPEECH-ENABLED SERVER FOR INTERNET WEBSITE AND METHOD

(75) Inventor: Ian M. Bennett, Palo Alto, CA (US)

(73) Assignee: Phoenix Solutions, Inc., Palo Alto, CA

(US)

(*) Notice: Subject to any disclaimer, the term of this

patent is extended or adjusted under 35

U.S.C. 154(b) by 0 days.

(21) Appl. No.: 09/439,174

(22) Filed: Nov. 12, 1999

(51) Int. Cl. G10L 15/02 (2006.01) G10L 15/18 (2006.01) G10L 15/22 (2006.01) G06F 17/20 (2006.01)

(52) **U.S. Cl.** **704/270.1**; 704/275; 707/3

(58) **Field of Classification Search** 704/231–235, 704/251, 255, 257, 260, 270, 270.1, 275; 707/3, 4, 5

See application file for complete search history.

(56) References Cited

U.S. PATENT DOCUMENTS

4,785,408	A	11/1988	Britton et al.
4,852,170	A	7/1989	Bordeaux
5,371,901	A	12/1994	Reed et al.
5,509,104	A	4/1996	Lee et al.
5,553,119	A	9/1996	McAlliser et al.
5,652,897	A	7/1997	Linebarger et al.
5,668,854	A	9/1997	Minakami et al.
5,675,707	A	10/1997	Gorin et al.
5,680,511	A	10/1997	Baker et al.
5,758,322	A	5/1998	Rongley 704/275
5,802,256	A	9/1998	Fawcett et al.
5,819,220	A	10/1998	Sarukkai et al 704/243
5,836,771	Α	11/1998	Ho et al.
5,860,063	A	1/1999	Gorin et al.
5,867,817	A	2/1999	Catallo et al 704/255
5,873,062	A	2/1999	Hansen et al.

(10) Patent No.: US 7,050,977 B1

(45) **Date of Patent:** May 23, 2006

5,884,3	02 A	3/1999	Но	
5,915,2	36 A	6/1999	Gould et al	704/251
5,934,9	10 A	8/1999	Ho et al.	
5,956,6	83 A	9/1999	Jacobs et al	704/275
5,960,3	94 A	9/1999	Gould et al	704/240
5,960,3	99 A	9/1999	Barclay et al	704/270

(Continued)

FOREIGN PATENT DOCUMENTS

EP 1 094 388 A2 4/2001

(Continued)

OTHER PUBLICATIONS

Arons, B., "The Design of Audio Servers and Toolkits for Supporting Speech in the User Interface," believed to be published in: Journal of the American Voice I/O Society, pp. 27-41, Mar. 1991.

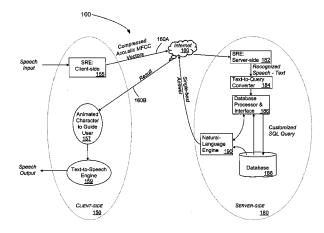
(Continued)

Primary Examiner—Martin Lerner (74) Attorney, Agent, or Firm—J. Nicholas Gross

(57) ABSTRACT

An Internet-based server with speech support for enhanced interactivity is disclosed. This server hosts a server-side speech recognition engine and additional linguistic and database functions that cooperate to provide enhanced interactivity for clients so that their browsing experience is more satisfying, efficient and productive. This human-like interactivity which allows the user to ask queries about topics that range from customer delivery, product descriptions, payment details, is facilitated by the allowing the user to articulate the his or her questions directly in his or her natural language. The answer typically provided in real-time, can also be interfaced and integrated with existing telephone, e-mail and other mixed media services to provide a single point of interactivity for the user when browsing at a web-site.

78 Claims, 31 Drawing Sheets



US 7,050,977 B1 Page 2

U.S. PATENT	DOCUMENTS	6,940,953 B1* 9/2005 Eberle et al
5,978,756 A 11/1999	Walker et al.	6,941,273 B1* 9/2005 Loghmani et al
5,987,410 A 11/1999	Kellner et al.	6,964,012 B1 * 11/2005 Zirngibl et al
, ,	Kendall et al.	6,965,864 B1 * 11/2005 Thrift et al
	Nguyen et al.	6,965,890 B1 * 11/2005 Dey et al 707/4
	Ramaswamy et al. Gillick et al 704/200	2001/0016813 A1 8/2001 Brown et al.
6,029,124 A 2/2000 6,032,111 A 2/2000		2001/0032083 A1 10/2001 Van Cleven
, ,	Brode et al.	2001/0056346 A1 12/2001 Ueyama et al. 2002/0032566 A1 3/2002 Tzirkel-Hancock et al.
6,044,266 A 3/2000	Kato	2002/0032566 A1 3/2002 Tzirkel-Hancock et al. 2002/0046023 A1 4/2002 Fujii et al.
	Gorin et al.	2002/0059068 A1 5/2002 Rose et al.
	Giangarra et al 704/275 Goldenthal et al.	2002/0059069 A1 5/2002 Hsu et al.
, ,	Kuhn et al.	2002/0086269 A1 7/2002 Shpiro
-,,	Moore et al.	2002/0087325 A1 7/2002 Lee et al. 2002/0087655 A1 7/2002 Bridgman et al.
	Raud et al.	2002/0091527 A1 7/2002 Shiau
	Guberman 704/207	2003/0191625 A1 10/2003 Gorin et al.
6,141,640 A 10/2000	Moo Walsh et al.	2005/0091056 A1 4/2005 Surace et al.
, ,	Surace et al 704/257	2005/0131704 A1 6/2005 Dragosh et al.
	Hambleton et al.	FOREIGN PATENT DOCUMENTS
-,,	Balakrishnan et al.	EP 1 096 471 A1 5/2001
, , , , , , , , , , , , , , , , , , ,	Culliss	WO WO 99/48011 A1 9/1999
, , , , , , , , , , , , , , , , , , ,	Hedin et al. Abella et al.	WO WO 99/50830 10/1999
, , ,	Crupi et al.	WO WO 00/14727 A1 3/2000
	Keiller et al.	WO WO 00/17854 A1 3/2000
-,,	Balakrishnan	WO WO 00/20962 A2 4/2000 WO WO 00/21075 A1 4/2000
· · · · · · · · · · · · · · · · · · ·	Mohri et al.	WO WO 00/21232 A2 4/2000
' ' '	Ammicht et al. Polcyn	WO WO 00/22610 A1 4/2000
	Digalakis et al.	WO WO 00/30072 A2 5/2000
	Ladd et al.	WO WO 00/30287 A1 5/2000 WO WO 00/68823 A2 11/2000
	Chung et al.	WO WO 00/08823 A2 11/2000 WO WO 01/16936 A1 3/2001
, ,	Jackson et al. Urs et al.	WO WO 01/18693 A2 3/2001
- , ,	Smith et al.	WO WO 01/26093 A1 4/2001
6,327,568 B1 12/2001		WO WO 01/78065 A1 10/2001
	Horiguchi et al 704/4	WO WO 01/95312 A1 12/2001 WO WO 02/03380 A1 1/2002
	Urs et al.	
	Hunt et al.	OTHER PUBLICATIONS
, ,	Eichstaedt et al.	Hazen, T et al., "Recent Improvements in an Approach to
	Meunier et al.	Segment-Based Automatic Language Identification,"
	White et al.	believed to be published in: Proceedings of the 1994 Inter-
	Chang Perrone 379/88.01	national Conference on Spoken Language Processing,
	Cook et al.	Yokohama, Japan, pp. 1883-1886, Sep. 1994.
	Walker et al.	House, D., "Spoken-Language Access to Multimedia (SLAM): A Multimodal Interface to the World-Wide Web,"
	Hughes et al.	Masters Thesis, Oregon Graduate Institute, Department of
6,499,011 B1 12/2002 6,499,013 B1 12/2002	Souvignier et al. Weber	Computer Science & Engineering, 59 pages, Apr. 1995.
	Norton et al.	Julia, L. et al., "http://www.speech. sri.com/demos/ atis.
	Ruber et al.	html," believed to be published in: Proceedings AAAI'97:
6,522,725 B1 2/2003		Stanford, pp. 72-76, Jul. 1997.
	Weber Ladd et al 704/275	Lau, R. et al, "Webgalaxy-Integrating Spoken Language and
	Chao Chang et al.	Hypertext Navigation," believed to be published in: in
	Mohri et al.	Kokkinakis, G. et al., (Eds.) Eurospeech '97, Proceedings of
	Warthen	the 5th European Conference on Speech Communication
	Polcyn Bijretrom et al	and Technology, Rhodes (Greece), Sep. 22-25, 1997: pp. 883-886, 1997.
	Bjurstrom et al. Polcyn	Digalakis, V. et al., "Product-Code Vector Quantization of
	Colbath et al 707/6	Cepstral Parameters for Speech Recognition over the
6,633,846 B1* 10/2003	Bennett et al 704/257	WWW," believed to be published in: Proc. ICSLP '98, 4
	Gorin et al.	pages, 1998.
	Beutnagel et al. Halverson et al 709/218	Melin, H., "On Word Boundary Detection in Digit-Based
	Keiller et al.	Speaker Verification," believed to be published in: Work-
6,871,179 B1 3/2005	Kist et al.	shop on Speaker Recognition and Its Commercial and
	Kuhn et al	Forensic Applications (RLA2C), Avignon, France, Apr.
6,922,733 B1* 7/2005	Kuiken et al 709/246	20-23, pp. 46-49, 1998.

US 7,050,977 B1

Page 3

Ramaswamy, G. et al., "Compression of Acoustic Features for Speech Recognition in Network Environments," believed to be published in: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 977-980, Jun. 1998.

Lu, B. et al., "Scalability Issues in the Real Time Protocol (RTP)," Project Report for CPSC 663 (Real Time Systems), Dept. of Computer Science, Texas A & M University, 19 pages, 1999.

Giuliani, D. et al., "Training of HMM with Filtered Speech Material for Hands-Free Recognition," believed to be published in: Proceedings of ICASSP '99, Phoenix, USA, 4 pages, 1999.

Digalakis, V. et al., "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web," believed to be published in: IEEE Journal on Selected Areas of Communications, 22 pages, 1999.

Tsakalidis, S. et al., "Efficient Speech Recognition Using Subvector Quantization and Discrete-Mixture HMMs," believed to be published in: Proc. ICASSP '99, 4 pages, 1999

Lin, B. et al., "A Distributed Architecture for Cooperative Spoken Dialogue Agents with Coherent Dialogue State and History," believed to be published in: IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado, USA, 4 pages, Dec. 1999.

Meunier, J., "RTP Payload Format for Distributed Speech Recognition," 48th IETF AVT WG—Aug. 3, 2000, 10 pages, 2000.

Sand Cherry Networks, SoftServer product literature, 2 pages 2001.

Kim,H. et al., "A Bitstream-Based Front-End for Wireless Speech Recognition on IS-136 Communications System," IEEE Transactions on Speech and Audio Processing, vol. 9, No. 5, pp. 558-568, Jul. 2001.

Agarwal, R., Towards a PURE Spoken Dialogue System for Information Access, believed to be published in *Proceedings* of the ACL/EACL Workshop on Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications, Madrid, Spain, 1997, 9 pages.

Ammicht, Egbert et al., "Knowledge Collection for Natural Language Spoken Dialog Systems," believed to be published in *Proc. Eurospeech*, vol. 3, p. 1375-1378, Budapest, Hungary, Sep. 1999, 4 pages.

AT&T Corp., "Network Watson 1.0 System Overview," 1998, 4 pages.

AT&T Corp., "AT&T Watson Advanced Speech Applications Platform," 1996, 3 pages.

AT&T Corp., "AT&T Watson Advanced Speech Applications Platform Version 2.0," 1996, 3 pages.

AT&T Corp., "AT&T Watson Advanced Speech Application Platform Version 2.0," 1996, 8 pages.

Gorin, Allen, "Processing of Semantic Information in Fluently Spoken Language," believed to be published in *Proc. ICSLP*, Philadelphia, PA, Oct. 1996, 4 pages.

Gorin, Allen et al., "How May I Help You," believed to be published in *Proc. IVTTA*, Basking Ridge, NJ, Oct. 1996, 32 pages.

Mohru, Mehryar, "String Matching With Automata," Nordic Journal of Computing, 1997, 15 pages.

Prudential News, "Prudential Pilots Revolutionary New Speech-Based Telephone Customer Service System Developed by AT&T Labs—Company Business and Marketing," Dec. 6, 1999, 3 pages.

Riccardi, Giuseppe et al., "A spoken language system for automated call routing," believed to be published in *Proc. ICASSP '97*, 1997, 4 pages.

Sharp, Douglas, et al., "The Watson Speech Recognition Engine," accepted by ICASSP, 1997, 9 pages.

European Patent Office search report for EP Application No. 00977144, dated Mar. 30, 2005, 5 pages.

Burstein, A. et al. "Using Speech Recognition In A Personal Communications System," *Proceedings of the International Conference on Communications*; Chicago, Illinois, Jun. 14-18, 1992, pp. 1717-1721.

Digalakis, V. et al., "Quantization Of Cepstral Parameters For Speech Recognition Over The World Wide Web," *IEEE Journal on Selected Areas of Communications*, 1999, pp. 82-90

Kuhn, T. et al., "Hybrid In-Car Speech Recognition For Mobile Multimedia Applications," *Vehicular Technology Conference*, Houston, Texas, May 1999, pp. 2009-2013.

L. Travis, "Handbook of Speech Pathology", Appleton-Century-Crofts, Inc., 1957, pp. 91-12-4.

L.E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains", Ann. Math. Stat., 37: 1554-1563, 1966.

J.L. Flanagan, "Speech Analysis Synthesis and Perception", 2nd edition, Springer-Verlag Berlin, 1972, pp. 1-53.

L.E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1-8, 1972.

Cox, Richard V. et al., "Speech and Language Processing for Next-Millennium Communications Services," Proceedings of the IEEE, vol. 88, No. 8, Aug. 2000, pp. 1314-1337.

Kuhn, Roland, and Renato De Mori, "The Application of Semantic Classification Trees to Natural Language Understanding," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, No. 5, May 1995, pp. 449-460.

Unisys Corp., "Natural Language Speech Assistant (NLSA) Capabilities Overview," NLR 3.0, Aug. 1998, Malvern, PA, 27 pages.

* cited by examiner

May 23, 2006

Sheet 1 of 31

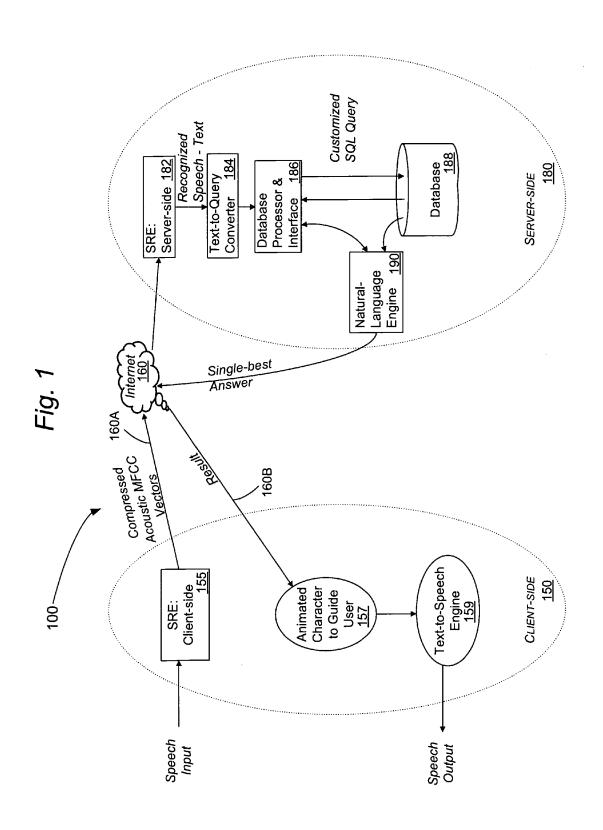


Figure 2 (Page 1/3)

CLIENT-SIDE SYSTEM LOGIC

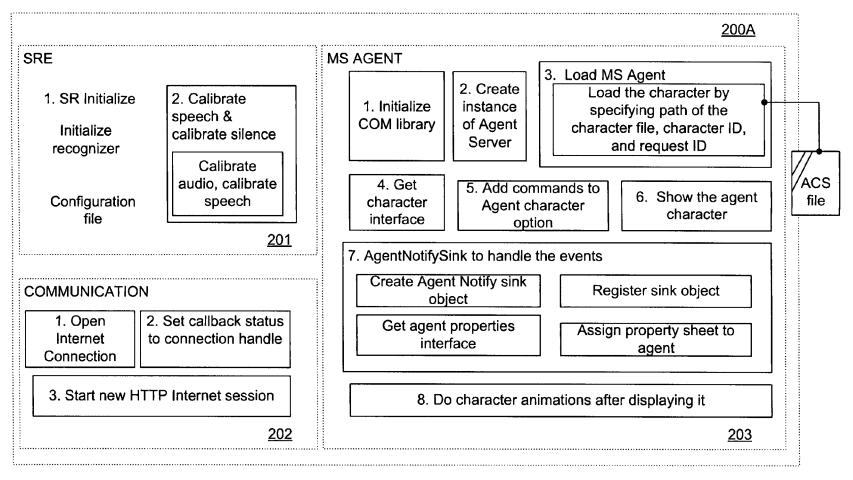


Figure 2 (Page 2/3)

U.S. Patent

May 23, 2006

Sheet 3 of 31

US 7,050,977 B1

CLIENT-SIDE SYSTEM LOGIC

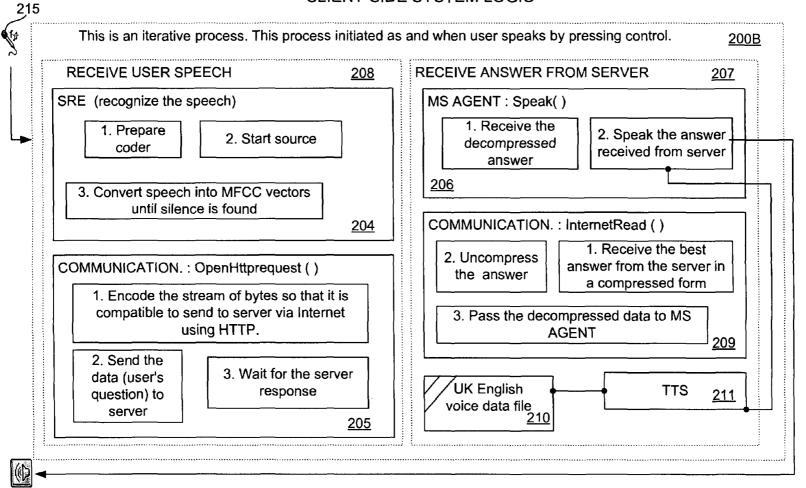


Figure 2 (Page 3/3)

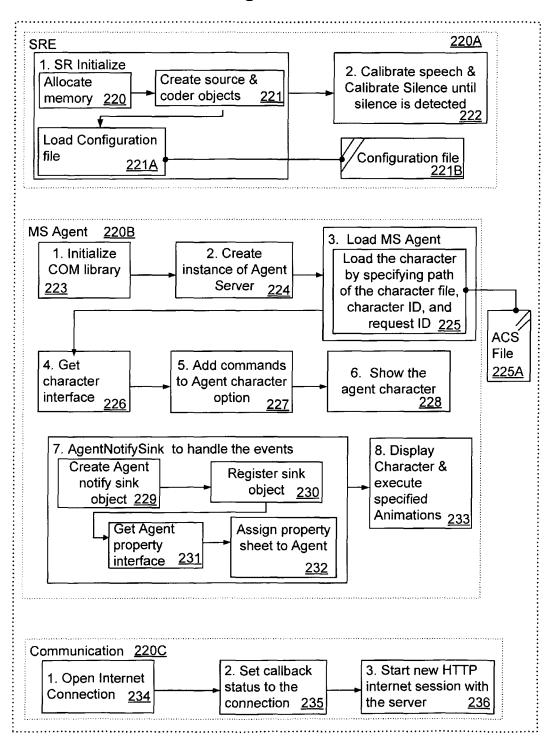
CLIENT-SIDE SYSTEM LOGIC

UN-INITIALIZATION (performs as and when user quits i.e. closes the web page) 200C COMMUNICATION 213 **MS AGENT** SRE 212 2. Release 1. Release 3. Unload 1. Close the Internet commands character agent handle i.e. the Interface Interface a. Delete the objects connection created while established with initialization process server. 5. Release 4. Release 6. Unregister b. Deallocate the AgentNotifysink prop. sheet **Agent Notifysink** memory assigned to Interface Interface the structure, which will be holding the 2. Close the Internet parameters for session which is speech created at the time of initialization 7. Release Agent Interface 214

May 23, 2006

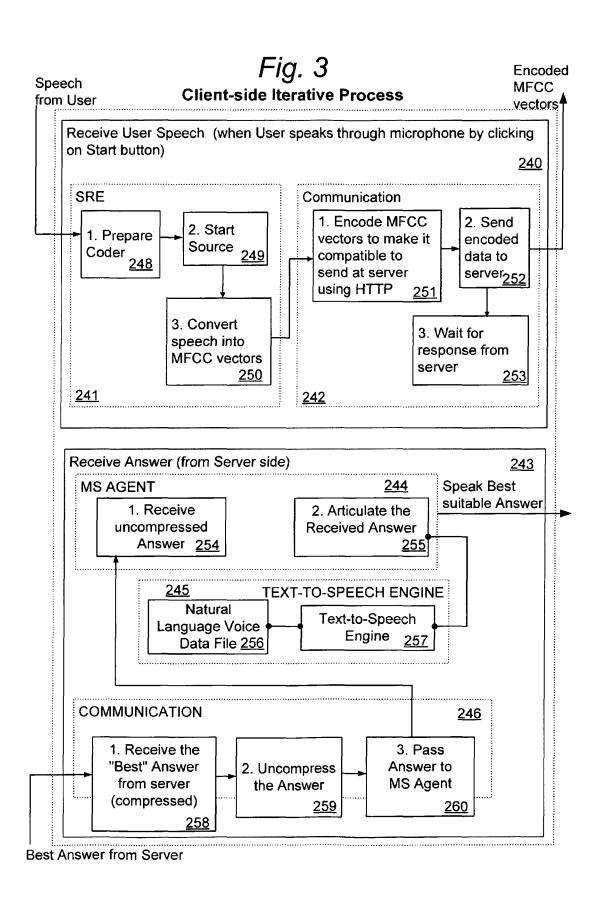
Sheet 5 of 31

Fig. 2-2 Client-side Initialization



May 23, 2006

Sheet 6 of 31



May 23, 2006

Sheet 7 of 31

Fig. 4
Client-side Un-Initialization

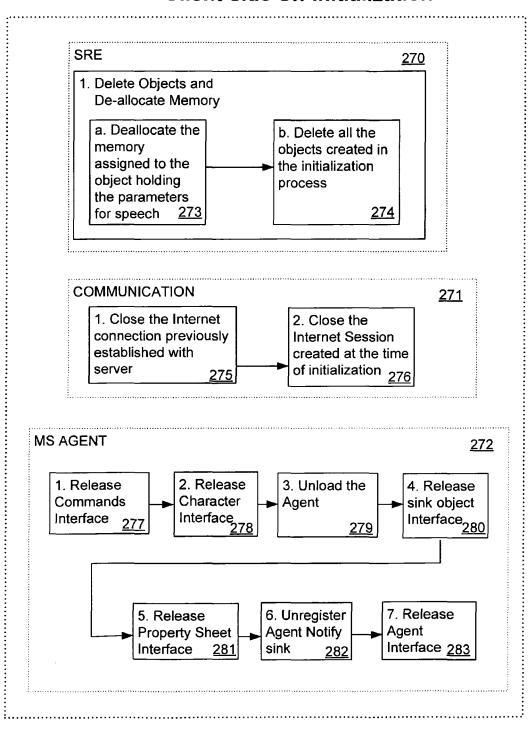
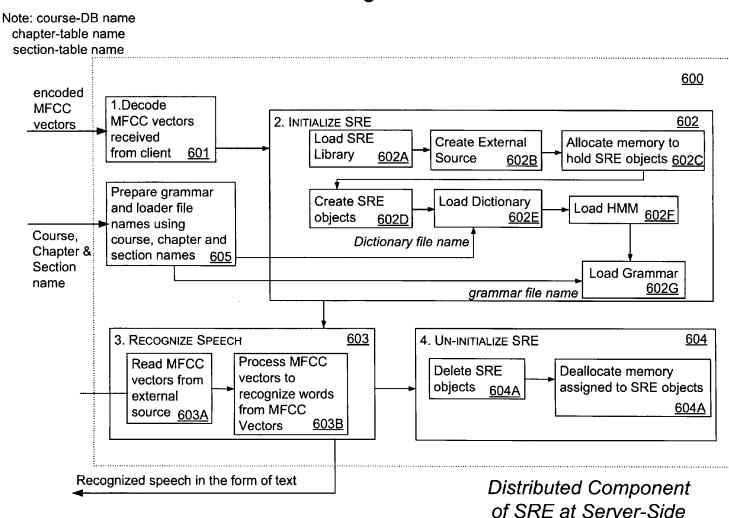
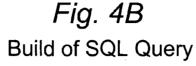
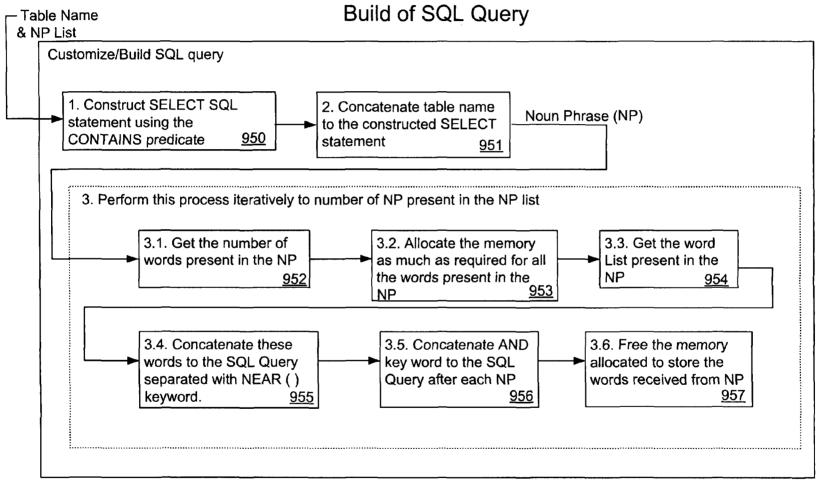


Fig. 4A

Document 52-4



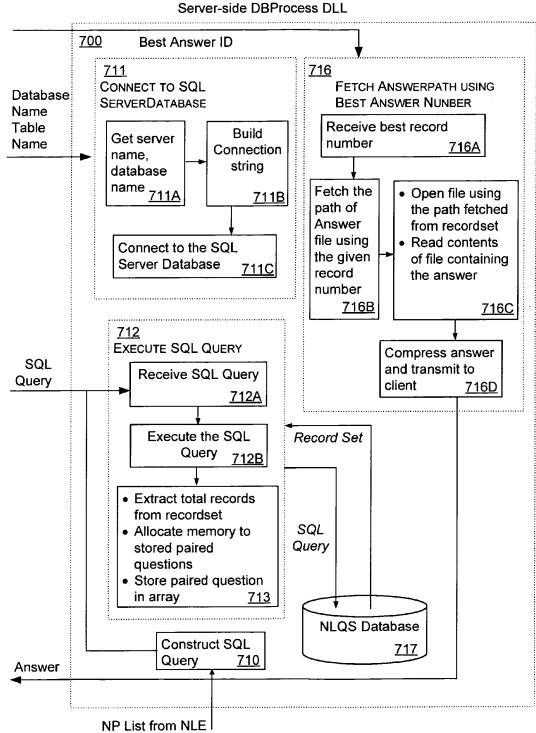




May 23, 2006

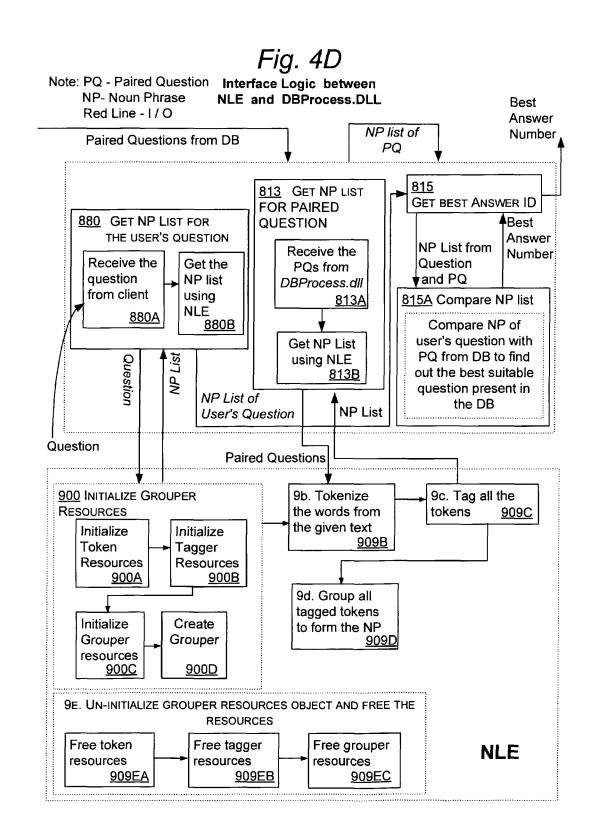
Sheet 10 of 31

Fig. 4C



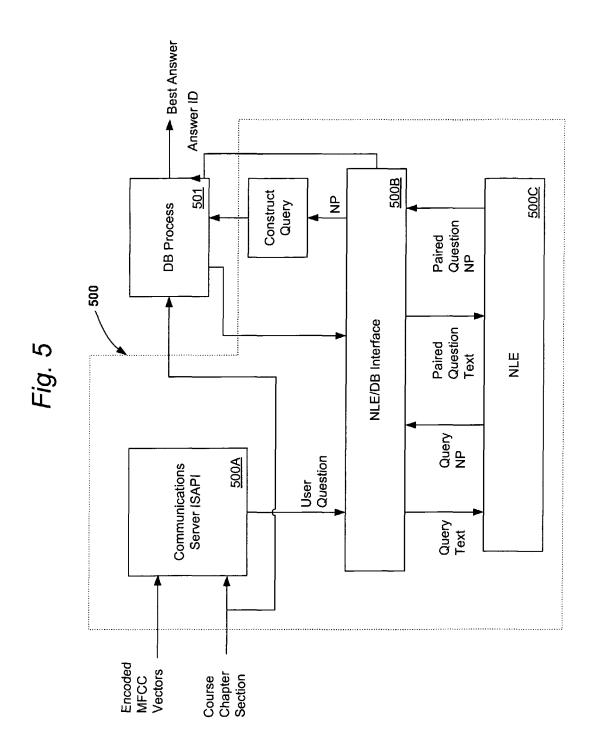
May 23, 2006

Sheet 11 of 31



May 23, 2006

Sheet 12 of 31



U.S. Patent May 23, 2006 Sheet 13 of 31 US 7,050,977 B1

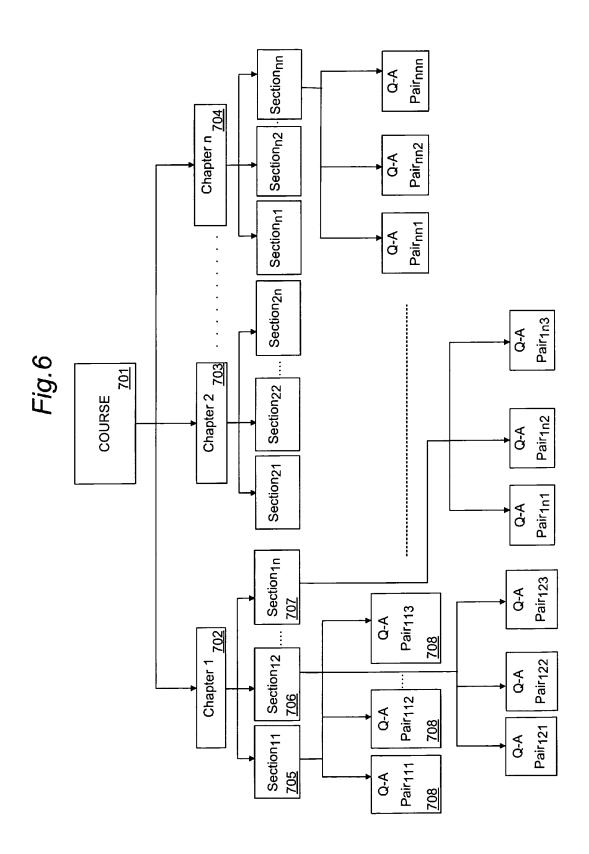


Fig.7A

FIELD NAME 701A	Data Type <u>702A</u>	Size <u>703A</u>	NULL <u>704A</u>	PRIMARY KEY <u>705A</u>	INDEXED? <u>706A</u>
ChapterName <u>707A</u>	Varchar	255	No	No	Yes
SectionName <u>708A</u>	Varchar	255	No	No	Yes

U.S. Patent May 23, 2006

Sheet 15 of 31

FIELD NAME 720	D ATA TYPE <u>721</u>	SizE 722	NULL 723	PRIMARY KEY	INDEXED?
Chapter_ID 726	Integer				Yes
Answer_ID 727	Char	5	No	UNIQUE	Yes
Section_Name	Varchar	255	ON.	UNIQUE	Yes
Answer_Title 729	Varchar	255	Yes	No	Yes
PairedQuestion 730	Text	16	No	O.	Yes (Full-Text)
AnswerPath	Varchar	255	No	No	Yes
Creator 732	Varchar	50	No	ON.	Yes
Date_of_Creation	Date	•	No	No	Yes
Date_of_Modification	Date	1	o _N	ON.	Yes

Fig. 7C

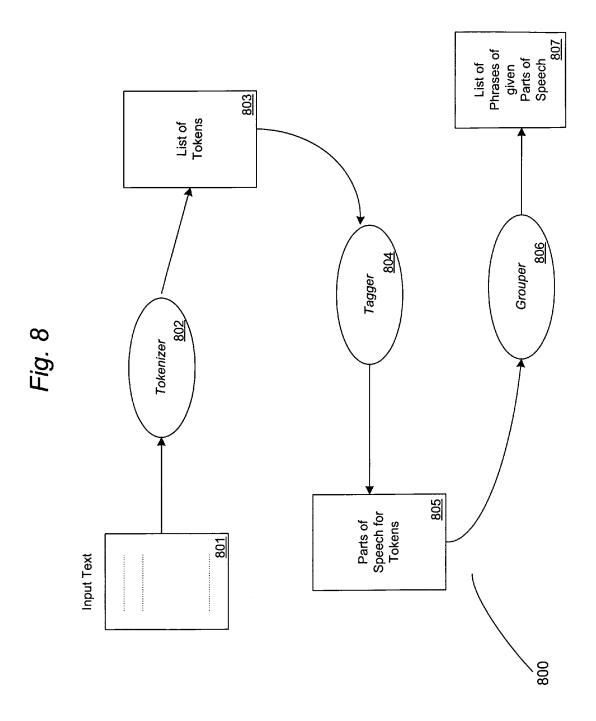
Document 52-4

Field		Description
720		735
AnswerID An integer - automatically incren		An integer - automatically incremented for user convenience
I Section Name		Name of section to which the particular record belongs. This field along with AnswerlD has to be made primary key
Answer_Title	<u>729</u>	A short description of the answer
PairedQuestion 730		Contains one or more combinations of questions for the related answer whose path is stored in the next column AnswerPath
I Answerpain		Contains the path of text file, which contains theanswer to the rleated qusetions stored in the previous column
Creator	<u>732</u>	Name of content creator
Date_of_Creation	<u>733</u>	Date on which content has been added
Date_of_Modification	<u>734</u>	Date on which content has been changed or modified

Fig. 7D

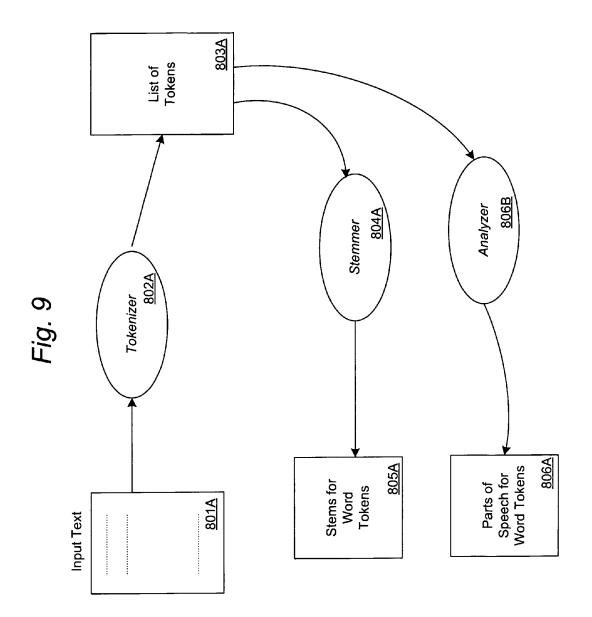
FIELD <u>740</u>	DATA TYPE <u>741</u>	Size <u>742</u>	NULL <u>743</u>	PRIMARY KEY <u>744</u>	INDEXED 745
Answer_ID 746	Char	5	No	Yes	Yes
Answer_Title 747	Varchar	255	Yes	No	No
PairedQuestion 748	Text	16	No	No	Yes (Full-Text)
Answer_Path 749	Varchar	255	No	No	No
Creator 750	Varchar	50	No	No	No
Date_of_Creation 751	Date	-	No	No	No
Date_of_Modification 752	Date	-	No	No	No

U.S. Patent May 23, 2006 Sheet 18 of 31 US 7,050,977 B1



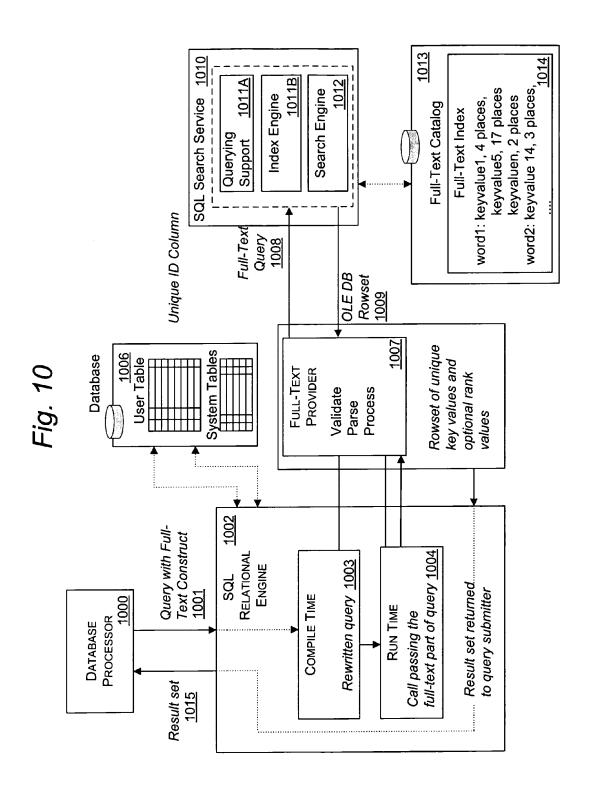
May 23, 2006

Sheet 19 of 31



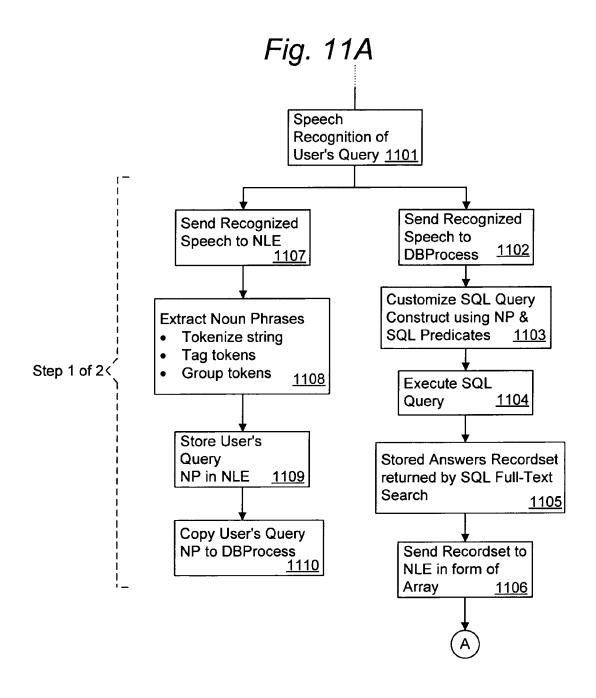
May 23, 2006

Sheet 20 of 31



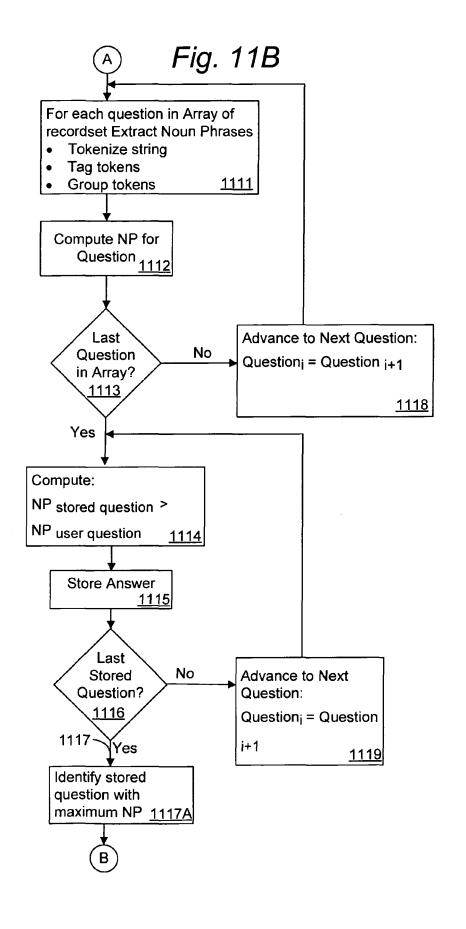
May 23, 2006

Sheet 21 of 31



May 23, 2006

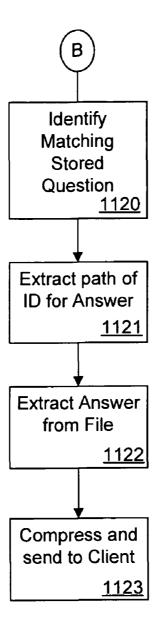
Sheet 22 of 31



May 23, 2006

Sheet 23 of 31

Fig. 11C



May 23, 2006

Sheet 24 of 31

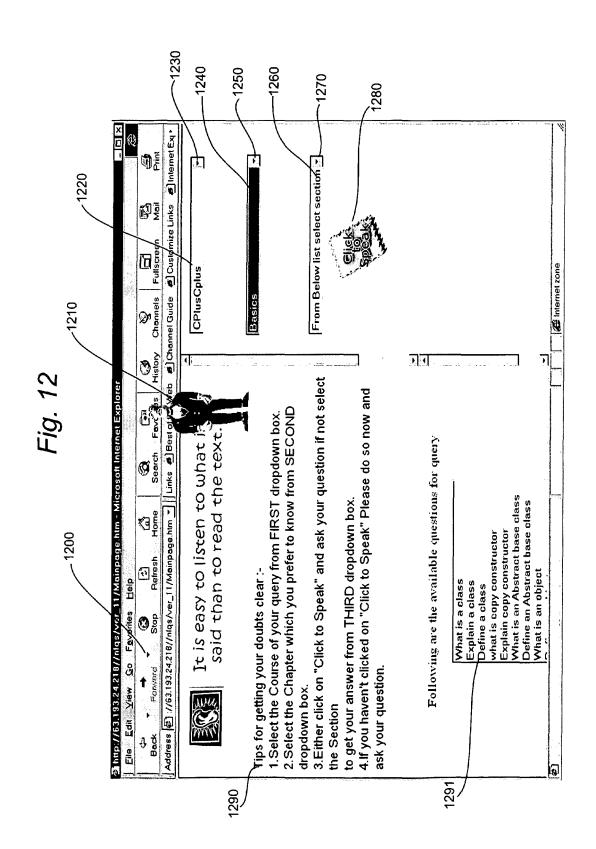
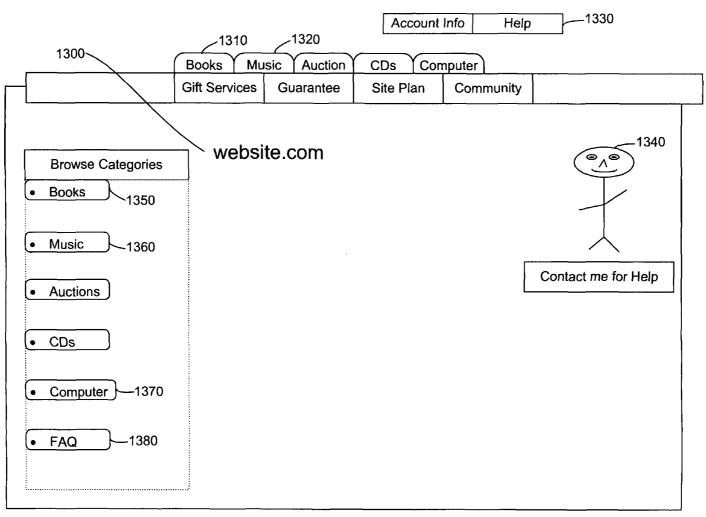
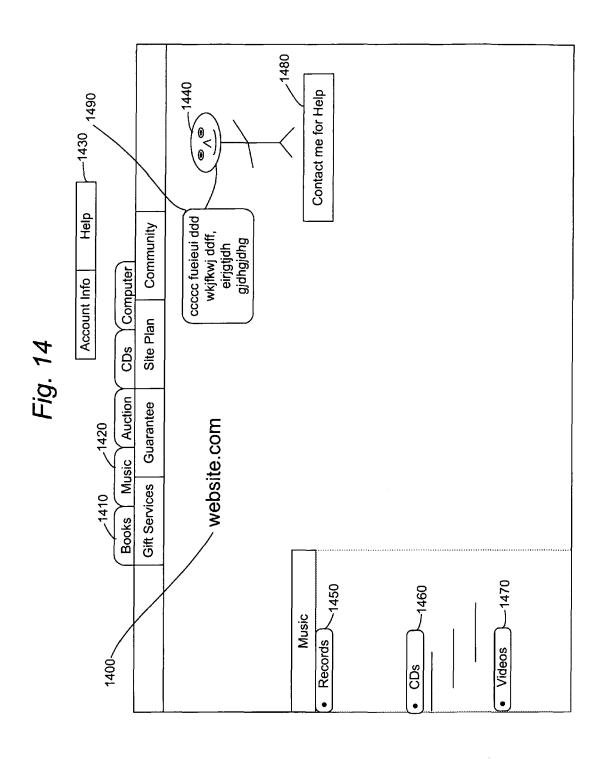


Fig. 13



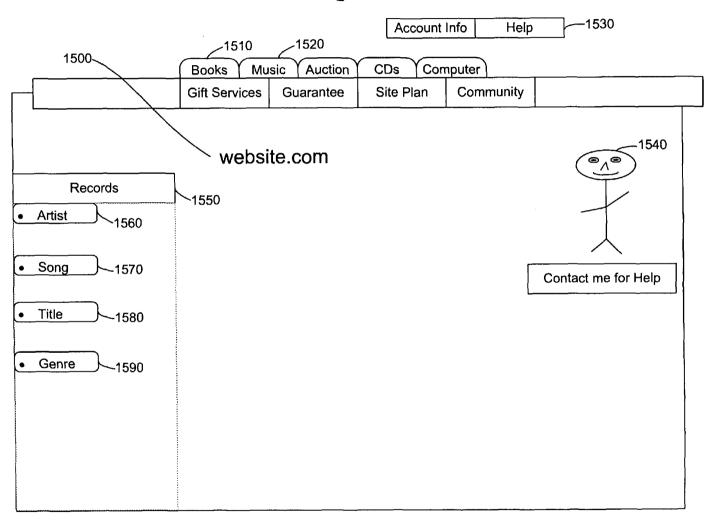
May 23, 2006

Sheet 26 of 31



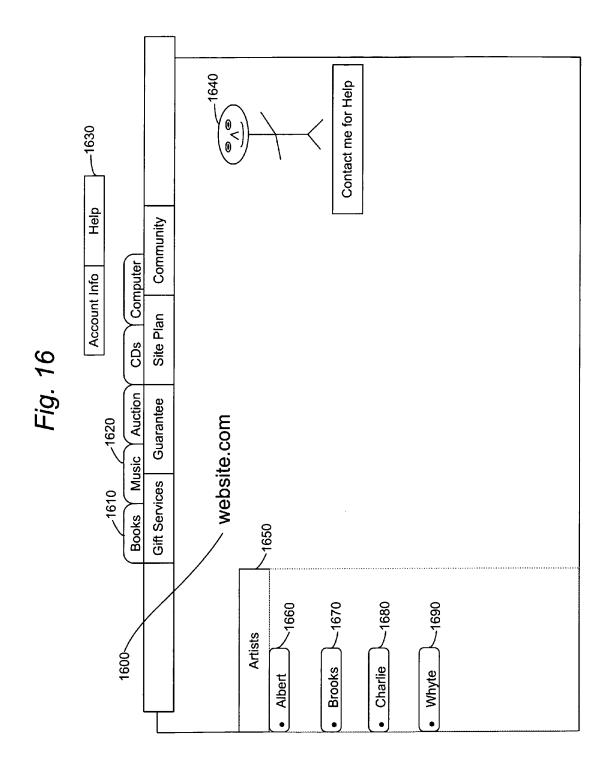
U.S. Patent

Fig. 15



May 23, 2006

Sheet 28 of 31



May 23, 2006

Sheet 29 of 31

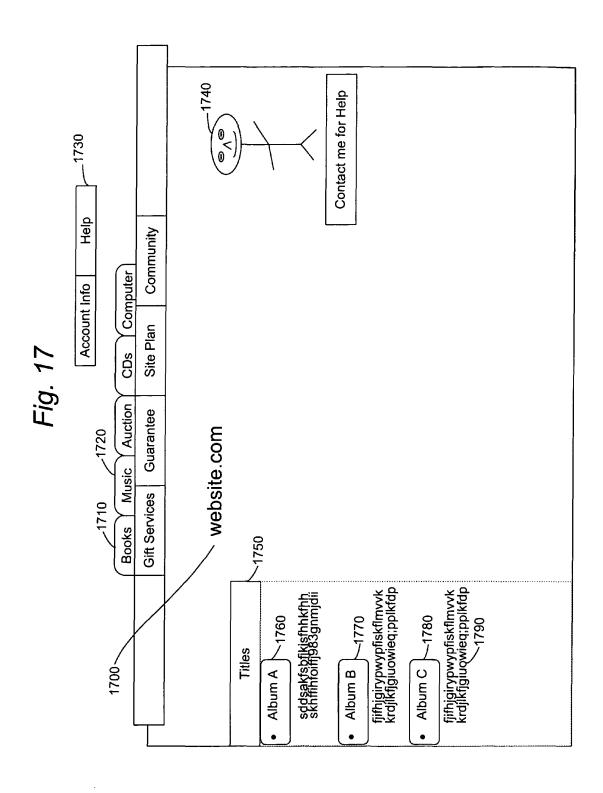


Fig. 18 (Page 1/2)

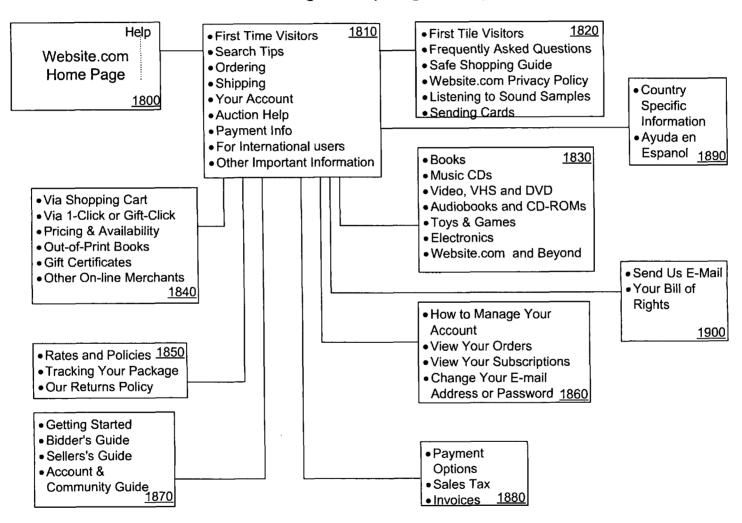
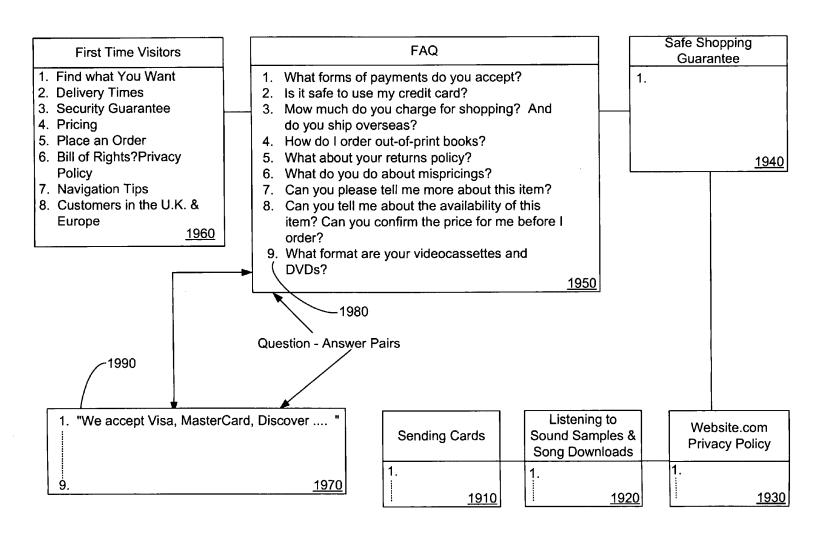


Fig. 18 (Page 2/2)



Case 3:08-cv-00863-MHP

SPEECH-ENABLED SERVER FOR INTERNET WEBSITE AND METHOD

RELATED APPLICATIONS

The present application is related to the following applications also filed contemporaneously herewith:

- 1) Ser. No. 09/439,145 entitled Distributed Real Time Speech Recognition System;
- 2) Ser. No. 09/439,173 entitled Speech Based Learning/ Training System;
- 3) Ser. No. 09/439,060 entitled Intelligent Ouerv Engine For Processing Voice Based Queries;

The above are incorporated by reference herein.

FIELD OF THE INVENTION

The invention relates to a system and an interactive method for an enabling a website to have interactive, real-20 time speech-enabled web pages. This interactive system is especially useful when implemented for e-commerce, e-support, search engines and the like, so that a user can intelligently and easily control an internet session using a conventional browser that is enhanced to handle speech 25 capabilities.

BACKGROUND OF THE INVENTION

The INTERNET, and in particular, the World-Wide Web (WWW), is growing in popularity and usage for both commercial and recreational purposes, and this trend is expected to continue. This phenomenon is being driven, in part, by the increasing and widespread use of personal computer systems and the availability of low cost INTER-NET access.

The emergence of inexpensive INTERNET access devices and high speed access techniques such as ADSL, cable modems, satellite modems, and the like, are expected to further accelerate the mass usage of the WWW.

Accordingly, it is expected that the number of entities offering services, products, etc., over the WWW will increase dramatically over the coming years. Until now, however, the INTERNET "experience" for users has been 45 limited mostly to non-voice based input/output devices, such as keyboards, intelligent electronic pads, mice, trackballs, printers, monitors, etc. This presents somewhat of a bottleneck for interacting over the WWW for a variety of reasons.

First, there is the issue of familiarity. Many kinds of 50 applications lend themselves much more naturally and fluently to a voice-based environment. For instance, most people shopping for audio recordings are very comfortable with asking a live sales clerk in a record store for information on titles by a particular author, where they can be found 55 in the store, etc. While it is often possible to browse and search on one's own to locate items of interest, it is usually easier and more efficient to get some form of human assistance first, and, with few exceptions, this request for assistance is presented in the form of a oral query. In 60 addition, many persons cannot or will not, because of physical or psychological barriers, use any of the aforementioned conventional I/O devices. For example, many older persons cannot easily read the text presented on WWW pages, or understand the layout/hierarchy of menus, or 65 manipulate a mouse to make finely coordinated movements to indicate their selections. Many others are intimidated by

the look and complexity of computer systems, WWW pages, etc., and therefore do not attempt to use online services for this reason as well.

Thus, applications which can mimic normal human interactions are likely to be preferred by potential on-line shoppers and persons looking for information over the WWW. It is also expected that the use of voice-based systems will increase the universe of persons willing to engage in e-commerce, e-learning, etc. To date, however, there are very few systems, if any, which permit this type of interaction, and, if they do, it is very limited. For example, various commercial programs sold by IBM (VIAVOICETM) and Kurzweil (DRAGONTM) permit some user control of the interface (opening, closing files) and searching (by using previously 15 trained URLs) but they do not present a flexible solution that can be used by a number of users across multiple cultures and without time consuming voice training. Typical prior efforts to implement voice based functionality in an INTER-NET context can be seen in U.S. Pat. No. 5,819,220 incorporated by reference herein.

Another issue presented by the lack of voice-based systems is efficiency. Many companies are now offering technical support over the INTERNET, and some even offer live operator assistance for such queries. While this is very advantageous (for the reasons mentioned above) it is also extremely costly and inefficient, because a real person must be employed to handle such queries. This presents a practical limit that results in long wait times for responses or high labor overheads. An example of this approach can be seen U.S. Pat. No. 5,802,526 also incorporated by reference herein. In general, a service presented over the WWW is far more desirable if it is "scalable," or, in other words, able to handle an increasing amount of user traffic with little if any perceived delay or troubles by a prospective user.

In a similar context, while remote learning has become an increasingly popular option for many students, it is practically impossible for an instructor to be able to field questions from more than one person at a time. Even then, such interaction usually takes place for only a limited period of time because of other instructor time constraints. To date, however, there is no practical way for students to continue a human-like question and answer type dialog after the learning session is over, or without the presence of the instructor to personally address such queries.

Conversely, another aspect of emulating a human-like dialog involves the use of oral feedback. In other words, many persons prefer to receive answers and information in audible form. While a form of this functionality is used by some websites to communicate information to visitors, it is not performed in a real-time, interactive question-answer dialog fashion so its effectiveness and usefulness is limited.

Yet another area that could benefit from speech-based interaction involves so-called "search" engines used by INTERNET users to locate information of interest at web sites, such as the those available at YAHOO®.com, METACRAWLER®.com, EXCITE®.com, etc. These tools permit the user to form a search query using either combinations of keywords or metacategories to search through a web page database containing text indices associated with one or more distinct web pages. After processing the user's request, therefore, the search engine returns a number of hits which correspond, generally, to URL pointers and text excerpts from the web pages that represent the closest match made by such search engine for the particular user query based on the search processing logic used by search engine. The structure and operation of such prior art search engines, including the mechanism by which they build the web page

1

database, and parse the search query, are well known in the art. To date, applicant is unaware of any such search engine that can easily and reliably search and retrieve information based on speech input from a user.

There are a number of reasons why the above environ- 5 ments (e-commerce, e-support, remote learning, INTER-NET searching, etc.) do not utilize speech-based interfaces, despite the many benefits that would otherwise flow from such capability. First, there is obviously a requirement that the output of the speech recognizer be as accurate as 10 possible. One of the more reliable approaches to speech recognition used at this time is based on the Hidden Markov Model (HMM)—a model used to mathematically describe any time series. A conventional usage of this technique is disclosed, for example, in U.S. Pat. No. 4,587,670 incorpo- 15 rated by reference herein. Because speech is considered to have an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated 20 with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. This finite state 25 machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density B₂(O₂). It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name 30 Hidden Markov Model. The basic theory of HMMs was published in a series of classic papers by Baum and his colleagues in the late 1960's and early 1970's. HMMs were first used in speech applications by Baker at Carnegie Mellon, by Jelenik and colleagues at IBM in the late 1970's 35 and by Steve Young and colleagues at Cambridge University, UK in the 1990's. Some typical papers and texts are as follows:

- L. E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains", Ann. 40 Math. Stat., 37: 1554–1563, 1966
- 2. L. E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1–8, 1072
- J. H. Baker, "The dragon system—An Overview", IEEE Trans. on ASSP Proc., ASSP-23(1): 24–29, February 1975
- F. Jeninek et al, "Continuous Speech Recognition: Statistical methods" in Handbook of Statistics, II, P. R. 50 Kristnaiad, Ed. Amsterdam, The Netherlands, North-Holland, 1982
- L. R. Bahl, F. Jeninek, R. L. Mercer, "A maximum likelihood approach to continuous speech recognition", IEEE Trans. Pattern Anal. Mach. Intell., PAMI-5: 55 179–190, 1983
- J. D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, N.J. 1980.
- H. R. Rabiner and B. H. Juang, "Fundamentals of 60 Speech Recognition", Prentice Hall, 1993
- H. R. Rabiner, "Digital Processing of Speech Signals", Prentice Hall, 1978

More recently research has progressed in extending HMM and combining HMMs with neural networks to speech 65 recognition applications at various laboratories. The following is a representative paper:

4

 Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco (1993), Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition. *Journal of Pattern Recognition and Artificial Intelligence*, Vol. 7, No. 4 pp. 899–916. Also in I. Guyon and P. Wang editors, *Advances in Pattern Recognition Systems using Neural Networks*, Vol. 7 of a Series in Machine Perception and Artificial Intelligence. World Scientific, February 1994.

All of the above are hereby incorporated by reference. While the HMM-based speech recognition yields very good results, contemporary variations of this technique cannot guarantee a word accuracy requirement of 100% exactly and consistently, as will be required for WWW applications for all possible all user and environment conditions. Thus, although speech recognition technology has been available for several years, and has improved significantly, the technical requirements have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily.

In contrast to word recognition, Natural language processing (NLP) is concerned with the parsing, understanding and indexing of transcribed utterances and larger linguistic units. Because spontaneous speech contains many surface phenomena such as disfluencies,—hesitations, repairs and restarts, discourse markers such as 'well' and other elements which cannot be handled by the typical speech recognizer, it is the problem and the source of the large gap that separates speech recognition and natural language processing technologies. Except for silence between utterances, another problem is the absence of any marked punctuation available for segmenting the speech input into meaningful units such as utterances. For optimal NLP performance, these types of phenomena should be annotated at its input. However, most continuous speech recognition systems produce only a raw sequence of words. Examples of conventional systems using NLP are shown in U.S. Pat. Nos. 4,991,094, 5,068,789, 5,146,405 and 5,680,628, all of which are incorporated by reference herein.

Second, most of the very reliable voice recognition systems are speaker-dependent, requiring that the interface be "trained" with the user's voice, which takes a lot of time, and is thus very undesirable from the perspective of a WWW environment, where a user may interact only a few times with a particular website. Furthermore, speaker-dependent systems usually require a large user dictionary (one for each unique user) which reduces the speed of recognition. This makes it much harder to implement a real-time dialog interface with satisfactory response capability (i.e., something that mirrors normal conversation—on the order of 3–5 seconds is probably ideal). At present, the typical shrinkwrapped speech recognition application software include offerings from IBM (VIAVOICETM) and Dragon Systems (DRAGONTM). While most of these applications are adequate for dictation and other transcribing applications, they are woefully inadequate for applications such as NLQS where the word error rate must be close to 0%. In addition these offerings require long training times and are typically are non client-server configurations. Other types of trained systems are discussed in U.S. Pat. No. 5,231,670 assigned to Kurzweil, and which is also incorporated by reference herein.

Another significant problem faced in a distributed voicebased system is a lack of uniformity/control in the speech recognition process. In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on

a single client. A well-known system of this type is depicted in U.S. Pat. No. 4,991,217 incorporated by reference herein. These clients can take numerous forms (desktop PC, laptop PC, PDA, etc.) having varying speech signal processing and communications capability. Thus, from the server side per- 5 spective, it is not easy to assure uniform treatment of all users accessing a voice-enabled web page, since such users may have significantly disparate word recognition and error rate performances. While a prior art reference to Gould et al.—U.S. Pat. No. 5,915,236—discusses generally the notion of tailoring a recognition process to a set of available computational resources, it does not address or attempt to solve the issue of how to optimize resources in a distributed environment such as a client-server model. Again, to enable such voice-based technologies on a wide-spread scale it is far more preferable to have a system that harmonizes and accounts for discrepancies in individual systems so that even the thinnest client is supportable, and so that all users are able to interact in a satisfactory manner with the remote server running the e-commerce, e-support and/or remote 20 learning application.

Two references that refer to a distributed approach for speech recognition include U.S. Pat. Nos. 5,956,683 and 5,960,399 incorporated by reference herein. In the first of these, U.S. Pat. No. 5,956,683—Distributed Voice Recognition System (assigned to Qualcomm) an implementation of a distributed voice recognition system between a telephony-based handset and a remote station is described. In this implementation, all of the word recognition operations seem to take place at the handset. This is done since the patent describes the benefits that result from locating of the system for acoustic feature extraction at the portable or cellular phone in order to limit degradation of the acoustic features due to quantization distortion resulting from the narrow bandwidth telephony channel. This reference therefore does not address the issue of how to ensure adequate performance for a very thin client platform. Moreover, it is difficult to determine, how, if at all, the system can perform real-time word recognition, and there is no meaningful description of how to integrate the system with a natural language processor.

The second of these references—U.S. Pat. No. 5,960, 399—Client/Server Speech Processor/Recognizer (assigned to GTE) describes the implementation of a HMM-based distributed speech recognition system. This reference is not instructive in many respects, however, including how to optimize acoustic feature extraction for a variety of client platforms, such as by performing a partial word recognition process where appropriate. Most importantly, there is only a 50 description of a primitive server-based recognizer that only recognizes the user's speech and simply returns certain keywords such as the user's name and travel destination to fill out a dedicated form on the user's machine. Also, the streaming of the acoustic parameters does not appear to be 55 implemented in real-time as it can only take place after silence is detected. Finally, while the reference mentions the possible use of natural language processing (column 9) there is no explanation of how such function might be implemented in a real-time fashion to provide an interactive feel for the user.

SUMMARY OF THE INVENTION

An object of the present invention, therefore, is to provide 65 an improved system and method for overcoming the limitations of the prior art noted above;

6

A primary object of the present invention is to provide a word and phrase recognition system that is flexibly and optimally distributed across a client/platform computing architecture, so that improved accuracy, speed and uniformity can be achieved for a wide group of users;

A further object of the present invention is to provide a speech recognition system that efficiently integrates a distributed word recognition system with a natural language processing system, so that both individual words and entire speech utterances can be quickly and accurately recognized in any number of possible languages;

A related object of the present invention is to provide an efficient query response system so that an extremely accurate, real-time set of appropriate answers can be given in response to speech-based queries;

Yet another object of the present invention is to provide an interactive, real-time instructional/learning system that is distributed across a client/server architecture, and permits a real-time question/answer session with an interactive character;

A related object of the present invention is to implement such interactive character with an articulated response capability so that the user experiences a human-like interaction;

Still a further object of the present invention is to provide an INTERNET website with speech processing capability so that voice based data and commands can be used to interact with such site, thus enabling voice-based e-commerce and e-support services to be easily scaleable;

Another object is to implement a distributed speech recognition system that utilizes environmental variables as part of the recognition process to improve accuracy and speed:

A further object is to provide a scaleable query/response database system, to support any number of query topics and users as needed for a particular application and instantaneous demand:

Yet another object of the present invention is to provide a query recognition system that employs a two-step approach, including a relatively rapid first step to narrow down the list of potential responses to a smaller candidate set, and a second more computationally intensive second step to identify the best choice to be returned in response to the query from the candidate set:

A further object of the present invention is to provide a natural language processing system that facilitates query recognition by extracting lexical components of speech utterances, which components can be used for rapidly identifying a candidate set of potential responses appropriate for such speech utterances;

Another related object of the present invention is to provide a natural language processing system that facilitates query recognition by comparing lexical components of speech utterances with a candidate set of potential response to provide an extremely accurate best response to such query.

One general aspect of the present invention, therefore, relates to a natural language query system (NLQS) that offers a fully interactive method for answering user's questions over a distributed network such as the INTERNET or a local intranet. This interactive system when implemented over the worldwide web (WWW) services of the INTERNET functions so that a client or user can ask a question in a natural language such as English, French, German or Spanish and receive the appropriate answer at his or her personal computer also in his or her native natural language.

The system is distributed and consists of a set of integrated software modules at the client's machine and another Document 52-4

set of integrated software programs resident on a server or set of servers. The client-side software program is comprised of a speech recognition program, an agent and its control program, and a communication program. The server-side program is comprised of a communication program, a natural language engine (NLE), a database processor (DBProcess), an interface program for interfacing the DBProcess with the NLE, and a SQL database. In addition, the client's machine is equipped with a microphone and a speaker. Processing of the speech utterance is divided between the client and server side so as to optimize processing and transmission latencies, and so as to provide support for even very thin client platforms.

In the context of an interactive learning application, the 15system is specifically used to provide a single-best answer to a user's question. The question that is asked at the client's machine is articulated by the speaker and captured by a microphone that is built in as in the case of a notebook computer or is supplied as a standard peripheral attachment. 20 Once the question is captured, the question is processed partially by NLQS client-side software resident in the client's machine. The output of this partial processing is a set of speech vectors that are transported to the server via the INTERNET to complete the recognition of the user's questions. This recognized speech is then converted to text at the

After the user's question is decoded by the speech recognition engine (SRE) located at the server, the question is converted to a structured query language (SQL) query. This query is then simultaneously presented to a software process within the server called DBProcess for preliminary processing and to a Natural Language Engine (NLE) module for extracting the noun phrases (NP) of the user's question. During the process of extracting the noun phrase within the NLE, the tokens of the users' question are tagged. The tagged tokens are then grouped so that the NP list can be determined. This information is stored and sent to the DBProcess process.

In the DBProcess, the SQL query is fully customized using the NP extracted from the user's question and other environment variables that are relevant to the application. For example, in a training application, the user's selection of course, chapter and or section would constitute the environ- 45 ment variables. The SQL query is constructed using the extended SQL Full-Text predicates -CONTAINS, FREE-TEXT, NEAR, AND. The SQL query is next sent to the Full-Text search engine within the SQL database, where a Full-Text search procedure is initiated. The result of this search procedure is recordset of answers. This recordset contains stored questions that are similar linguistically to the user's question. Each of these stored questions has a paired answer stored in a separate text file, whose path is stored in a table of the database.

The entire recordset of returned stored answers is then returned to the NLE engine in the form of an array. Each stored question of the array is then linguistically processed sequentially one by one. This linguistic processing constitutes the second step of a 2-step algorithm to determine the 60 single best answer to the user's question. This second step proceeds as follows: for each stored question that is returned in the recordset, a NP of the stored question is compared with the NP of the user's question. After all stored questions of the array are compared with the user's question, the stored 65 question that yields the maximum match with the user's question is selected as the best possible stored question that

matches the user's question. The metric that is used to determine the best possible stored question is the number of noun phrases.

The stored answer that is paired to the best-stored question is selected as the one that answers the user's question. The ID tag of the question is then passed to the DBProcess. This DBProcess returns the answer which is stored in a file.

A communication link is again established to send the answer back to the client in compressed form. The answer once received by the client is decompressed and articulated to the user by the text-to-speech engine. Thus, the invention can be used in any number of different applications involving interactive learning systems, INTERNET related commerce sites, INTERNET search engines, etc.

Computer-assisted instruction environments often require the assistance of mentors or live teachers to answer questions from students. This assistance often takes the form of organizing a separate pre-arranged forum or meeting time that is set aside for chat sessions or live call-in sessions so that at a scheduled time answers to questions may be provided. Because of the time immediacy and the ondemand or asynchronous nature of on-line training where a student may log on and take instruction at any time and at any location, it is important that answers to questions be provided in a timely and cost-effective manner so that the user or student can derive the maximum benefit from the material presented.

This invention addresses the above issues. It provides the user or student with answers to questions that are normally channeled to a live teacher or mentor. This invention provides a single-best answer to questions asked by the student. The student asks the question in his or her own voice in the language of choice. The speech is recognized and the answer to the question is found using a number of technologies including distributed speech recognition, full-text search database processing, natural language processing and textto-speech technologies. The answer is presented to the user, as in the case of a live teacher, in an articulated manner by an agent that mimics the mentor or teacher, and in the language of choice—English, French, German, Japanese or other natural spoken language. The user can choose the agent's gender as well as several speech parameters such as pitch, volume and speed of the character's voice.

Other applications that benefit from NLQS are e-commerce applications. In this application, the user's query for a price of a book, compact disk or for the availability of any item that is to be purchased can be retrieved without the need to pick through various lists on successive web pages. Instead, the answer is provided directly to the user without any additional user input.

Similarly, it is envisioned that this system can be used to provide answers to frequently-asked questions (FAQs), and as a diagnostic service tool for e-support. These questions are typical of a give web site and are provided to help the 55 user find information related to a payment procedure or the specifications of, or problems experienced with a product/ service. In all of these applications, the NLQS architecture can be applied.

A number of inventive methods associated with these architectures are also beneficially used in a variety of INTERNET related applications.

Although the inventions are described below in a set of preferred embodiments, it will be apparent to those skilled in the art the present inventions could be beneficially used in many environments where it is necessary to implement fast, accurate speech recognition, and/or to provide a human-like dialog capability to an intelligent system.

Document 52-4

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a preferred embodiment of a natural language query system (NLQS) of the present invention, which is distributed across a client/server com- 5 puting architecture, and can be used as an interactive learning system, an e-commerce system, an e-support system, and the like;

FIG. 2 is a block diagram of a preferred embodiment of a client side system, including speech capturing modules, 10 partial speech processing modules, encoding modules, transmission modules, agent control modules, and answer/voice feedback modules that can be used in the aforementioned

FIG. 2—2 is a block diagram of a preferred embodiment 15 of a set of initialization routines and procedures used for the client side system of FIG. 2;

FIG. 3 is a block diagram of a preferred embodiment of a set of routines and procedures used for handling an iterated set of speech utterances on the client side system of FIG. 2, 20 transmitting speech data for such utterances to a remote server, and receiving appropriate responses back from such

FIG. 4 is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for 25 un-initializing the client side system of FIG. 2;

FIG. 4A is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a distributed component of a speech recognition module for the server side system of FIG. 5;

FIG. 4B is a block diagram of a preferred set of routines and procedures used for implementing an SQL query builder for the server side system of FIG. 5;

FIG. 4C is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a database control process module for the server side system of FIG. 5;

FIG. 4D is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a natural language engine that provides query formulation support, a query response module, and an interface to the database control process module for the server side system of FIG. 5;

FIG. 5 is a block diagram of a preferred embodiment of a server side system, including a speech recognition module to complete processing of the speech utterances, environmental and grammar control modules, query formulation modules, a natural language engine, a database control module, and a query response module that can be used in the aforementioned NLQS;

FIG. 6 illustrates the organization of a full-text database used as part of server side system shown in FIG. 5;

FIG. 7A illustrates the organization of a full-text database course table used as part of server side system shown in FIG. 55 5 for an interactive learning embodiment of the present invention;

FIG. 7B illustrates the organization of a full-text database chapter table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present 60 invention;

FIG. 7C describes the fields used in a chapter table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention;

FIG. 7D describes the fields used in a section table used 65 as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention;

10

FIG. 8 is a flow diagram of a first set of operations performed by a preferred embodiment of a natural language engine on a speech utterance including Tokenization, Tagging and Grouping;

FIG. 9 is a flow diagram of the operations performed by a preferred embodiment of a natural language engine on a speech utterance including stemming and Lexical Analysis

FIG. 10 is a block diagram of a preferred embodiment of a SQL database search and support system for the present invention:

FIGS. 11A-11C are flow diagrams illustrating steps performed in a preferred two step process implemented for query recognition by the NLQS of FIG. 2;

FIG. 12 is an illustration of another embodiment of the present invention implemented as part of a Web-based speech based learning/training System;

FIGS. 13–17 are illustrations of another embodiment of the present invention implemented as part of a Web-based e-commerce system;

FIG. 18 is an illustration of another embodiment of the present invention implemented as part of a voice-based Help Page for an E-Commerce Web Site.

DETAILED DESCRIPTION OF THE INVENTION

Overview

As alluded to above, the present inventions allow a user to ask a question in a natural language such as English, French, German, Spanish or Japanese at a client computing system (which can be as simple as a personal digital assistant or cell-phone, or as sophisticated as a high end desktop PC) and receive an appropriate answer from a remote server also in his or her native natural language. As such, the embodiment of the invention shown in FIG. 1 is beneficially used in what can be generally described as a Natural Language Query System (NLQS) 100, which is configured to interact on a real-time basis to give a human-like dialog capability/ experience for e-commerce, e-support, and e-learning applications.

The processing for NLQS 100 is generally distributed across a client side system 150, a data link 160, and a server-side system 180. These components are well known in the art, and in a preferred embodiment include a personal computer system 150, an INTERNET connection 160A, 160B, and a larger scale computing system 180. It will be understood by those skilled in the art that these are merely exemplary components, and that the present invention is by no means limited to any particular implementation or combination of such systems. For example, client-side system 150 could also be implemented as a computer peripheral, a PDA, as part of a cell-phone, as part of an INTERNETadapted appliance, an INTERNET linked kiosk, etc. Similarly, while an INTERNET connection is depicted for data link 160A, it is apparent that any channel that is suitable for carrying data between client system 150 and server system 180 will suffice, including a wireless link, an RF link, an IR link, a LAN, and the like. Finally, it will be further appreciated that server system 180 may be a single, large-scale system, or a collection of smaller systems interlinked to support a number of potential network users.

Initially speech input is provided in the form of a question or query articulated by the speaker at the client's machine or personal accessory as a speech utterance. This speech utterance is captured and partially processed by NLQS clientside software 155 resident in the client's machine. To

Document 52-4

11

facilitate and enhance the human-like aspects of the interaction, the question is presented in the presence of an animated character 157 visible to the user who assists the user as a personal information retriever/agent. The agent can also interact with the user using both visible text output on 5 a monitor/display (not shown) and/or in audible form using a text to speech engine 159. The output of the partial processing done by SRE 155 is a set of speech vectors that are transmitted over communication channel 160 that links the user's machine or personal accessory to a server or 10 servers via the INTERNET or a wireless gateway that is linked to the INTERNET as explained above. At server 180, the partially processed speech signal data is handled by a server-side SRE 182, which then outputs recognized speech text corresponding to the user's question. Based on this user 15 question related text, a text-to-query converter 184 formulates a suitable query that is used as input to a database processor 186. Based on the query, database processor 186 then locates and retrieves an appropriate answer using a customized SQL query from database 188. A Natural Lan- 20 guage Engine 190 facilitates structuring the query to database 188. After a matching answer to the user's question is found, the former is transmitted in text form across data link 160B, where it is converted into speech by text to speech engine 159, and thus expressed as oral feedback by animated 25 character agent 157.

Because the speech processing is broken up in this fashion, it is possible to achieve real-time, interactive, human-like dialog consisting of a large, controllable set of questions/answers. The assistance of the animated agent 157 30 further enhances the experience, making it more natural and comfortable for even novice users. To make the speech recognition process more reliable, context-specific grammars and dictionaries are used, as well as natural language processing routines at NLE 190, to analyze user questions 35 lexically. While context-specific processing of speech data is known in the art (see e.g., U.S. Pat. Nos. 5,960,394, 5,867, 817, 5,758,322 and 5,384,892 incorporated by reference herein) the present inventors are unaware of any such implementation as embodied in the present inventions. The 40 text of the user's question is compared against text of other questions to identify the question posed by the user by DB processor/engine (DBE) 186. By optimizing the interaction and relationship of the SR engines 155 and 182, the NLP routines 190, and the dictionaries and grammars, an 45 extremely fast and accurate match can be made, so that a unique and responsive answer can be provided to the user.

On the server side 180, interleaved processing further accelerates the speech recognition process. In simplified terms, the query is presented simultaneously both to NLE 50 190 after the query is formulated, as well as to DBE 186. NLE 190 and SRE 182 perform complementary functions in the overall recognition process. In general, SRE 182 is primarily responsible for determining the identity of the words articulated by the user, while NLE 190 is responsible 55 for the linguistic morphological analysis of both the user's query and the search results returned after the database

After the user's query is analyzed by NLE 190 some parameters are extracted and sent to the DBProcess. Addi- 60 tional statistics are stored in an array for the 2nd step of processing. During the 2^{nd} step of 2-step algorithm, the recordset of preliminary search results are sent to the NLE 160 for processing. At the end of this 2^{nd} step, the single question that matches the user's query is sent to the DBPro- 65 cess where further processing yields the paired answer that is paired with the single best stored question.

12

Thus, the present invention uses a form of natural language processing (NLP) to achieve optimal performance in a speech based web application system. While NLP is known in the art, prior efforts in Natural Language Processing (NLP) work nonetheless have not been well integrated with Speech Recognition (SR) technologies to achieve reasonable results in a web-based application environment. In speech recognition, the result is typically a lattice of possible recognized words each with some probability of fit with the speech recognizer. As described before, the input to a typical NLP system is typically a large linguistic unit. The NLP system is then charged with the parsing, understanding and indexing of this large linguistic unit or set of transcribed utterances. The result of this NLP process is to understand lexically or morphologically the entire linguistic unit as opposed to word recognition. Put another way, the linguistic unit or sentence of connected words output by the SRE has to be understood lexically, as opposed to just being "recognized".

As indicated earlier, although speech recognition technology has been available for several years, the technical requirements for the NLQS invention have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily. In realizing that even with the best of conditions, it might be not be possible to achieve the perfect 100% speech recognition accuracy that is required, the present invention employs an algorithm that balances the potential risk of the speech recognition process with the requirements of the natural language processing so that even in cases where perfect speech recognition accuracy is not achieved for each word in the query, the entire query itself is nonetheless recognized with sufficient accuracy.

This recognition accuracy is achieved even while meeting very stringent user constraints, such as short latency periods of 3 to 5 seconds (ideally—ignoring transmission latencies which can vary) for responding to a speech-based query, and for a potential set of 100-250 query questions. This quick response time gives the overall appearance and experience of a real-time discourse that is more natural and pleasant from the user's perspective. Of course, non-real time applications, such as translation services for example, can also benefit from the present teachings as well, since a centralized set of HMMs, grammars, dictionaries, etc., are maintained.

General Aspects of Speech Recognition Used in the Present Inventions

General background information on speech recognition can be found in the prior art references discussed above and incorporated by reference herein. Nonetheless, a discussion of some particular exemplary forms of speech recognition structures and techniques that are well-suited for NLQS 100 is provided next to better illustrate some of the characteristics, qualities and features of the present inventions.

Speech recognition technology is typically of two types speaker independent and speaker dependent. In speakerdependent speech recognition technology, each user has a voice file in which a sample of each potentially recognized word is stored. Speaker-dependent speech recognition systems typically have large vocabularies and dictionaries making them suitable for applications as dictation and text transcribing. It follows also that the memory and processor resource requirements for the speaker-dependent can be and are typically large and intensive.

Document 52-4

13

Conversely speaker-independent speech recognition technology allows a large group of users to use a single vocabulary file. It follows then that the degree of accuracy that can be achieved is a function of the size and complexity of the grammars and dictionaries that can be supported for a given 5 language. Given the context of applications for which NLQS, the use of small grammars and dictionaries allow speaker independent speech recognition technology to be implemented in NLQS.

The key issues or requirements for either type—speaker- 10 independent or speaker-dependent, are accuracy and speed. As the size of the user dictionaries increase, the speech recognition accuracy metric-word error rate (WER) and the speed of recognition decreases. This is so because the search time increases and the pronunciation match becomes 15 more complex as the size of the dictionary increases.

The basis of the NLQS speech recognition system is a series of Hidden Markov Models (HMM), which, as alluded to earlier, are mathematical models used to characterize any time varying signal. Because parts of speech are considered 20 to be based on an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probabil- 25 ity distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to an associated probability distribution. This finite state machine changes state once every time unit, and 30 each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_i(O_t)$. It is only the outcome, not the state which is visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.

In isolated speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word can each be described by a Markov model as follows:

$$O=o_1, o_2, \dots o_T \tag{1-1}$$

where o, is a speech vector observed at time t. The isolated word recognition then is to compute:

$$arg max\{P(w_i|O)\} \tag{1-2}$$

By using Bayes' Rule,

In the general case, the Markov model when applied to speech also assumes a finite state machine which changes state once every time unit and each time that a state j is 50 entered, a speech vector o, is generated from the probability density $b_i(o_t)$. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability aij.

For a state sequence X, the joint probability that O is 55 generated by the model M moving through a state sequence X is the product of the transition probabilities and the output probabilities. Only the observation sequence is known—the state sequence is hidden as mentioned before.

Given that X is unknown, the required likelihood is 60 computed by summing over all possible state sequences $X=x(1), x(2), x(3), \dots x(T)$, that is

$$P(O|M) = \sum \{a_{x(0) \ x(1)} \Pi b(x)(o_t) a_{x(t) \ x(t+1)} \}$$

Given a set of models M_i , corresponding to words W_i 65 equation 1-2 is solved by using 1-3 and also by assuming that:

14

 $P(O|w_i)=P(O|M_i)$

All of this assumes that the parameters $\{a_{ij}\}$ and $\{b_j(o_t)\}$ are known for each model M_i. This can be done, as explained earlier, by using a set of training examples corresponding to a particular model. Thereafter, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure. So if a sufficient number of representative examples of each word are collected, then a HMM can be constructed which simply models all of the many sources of variability inherent in real speech. This training is well-known in the art, so it is not described at length herein, except to note that the distributed architecture of the present invention enhances the quality of HMMs, since they are derived and constituted at the server side, rather than the client side. In this way, appropriate samples from users of different geographical areas can be easily compiled and analyzed to optimize the possible variations expected to be seen across a particular language to be recognized. Uniformity of the speech recognition process is also well-maintained, and error diagnostics are simplified, since each prospective user is using the same set of HMMs during the recognition process.

To determine the parameters of a HMM from a set of training samples, the first step typically is to make a rough guess as to what they might be. Then a refinement is done using the Baum-Welch estimation formulae. By these formulae, the maximum likelihood estimates of μ_i (where μ_i is mean vector and Σ_i is covariance matrix) is:

$$\mu_j = \sum_{t=1}^T L_j(t) o_t / [\sum_{t=1}^T L_j(t) o_t]$$

A forward-backward algorithm is next used to calculate the probability of state occupation L_i(t). If the forward probability $\alpha_i(t)$ for some model M with N states is defined as:

$$\alpha_j(t)=P(o_1,\ldots,o_p|x(t)=j|M)$$

This probability can be calculated using the recursion:

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha(t-1)a_{ij}\right]b_j(o_t)$$

40

45

Similarly the backward probability can be computed using the recursion:

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij} b_j(o_t + 1)(t+1)$$

Realizing that the forward probability is a joint probability and the backward probability is a conditional probability, the probability of state occupation is the product of the two probabilities:

$$\alpha j(t)\beta_j(t){=}P(O,\;x(t){=}j{\mid}M)$$

Hence the probability of being in state j at a time t is:

$$L_i(t)=1/P[\alpha_i(t)\beta_i(t)]$$

where
$$P=P(O|M)$$

To generalize the above for continuous speech recognition, we assume the maximum likelihood state sequence where the summation is replaced by a maximum operation. Thus for a given model M, let ϕ_i (t) represent the maximum likelihood of observing speech vectors o₁ to o, and being used in state j at time t:

$$\Phi_j(t) = \max\{\Phi_j(t)(t-1)\alpha_{ij}\}\beta_j(o_t)$$

Expressing this logarithmically to avoid underflow, this likelihood becomes:

$$\psi_i(t) = \max\{\psi_i(t-1) + \log(\alpha_{ii})\} + \log(b_i(o_t))$$

This is also known as the Viterbi algorithm. It can be visualized as finding the best path through a matrix where the vertical dimension represents the states of the HMM and

15

horizontal dimension represents frames of speech i.e. time. To complete the extension to connected speech recognition, it is further assumed that each HMM representing the underlying sequence is connected. Thus the training data for continuous speech recognition should consist of connected 5 utterances; however, the boundaries between words do not have to be known.

To improve computational speed/efficiency, the Viterbi algorithm is sometimes extended to achieve convergence by using what is known as a Token Passing Model. The token 10 passing model represents a partial match between the observation sequence o₁ to o₂ and a particular model, subject to the constraint that the model is in state j at time t. This token passing model can be extended easily to connected speech environments as well if we allow the sequence of HMMs to 15 be defined as a finite state network. A composite network that includes both phoneme-based HMMs and complete words can be constructed so that a single-best word can be recognized to form connected speech using word N-best extraction from the lattice of possibilities. This composite 20 form of HMM-based connected speech recognizer is the basis of the NLQS speech recognizer module. Nonetheless, the present invention is not limited as such to such specific forms of speech recognizers, and can employ other techniques for speech recognition if they are otherwise compat- 25 ible with the present architecture and meet necessary performance criteria for accuracy and speed to provide a real-time dialog experience for users.

The representation of speech for the present invention's HMM-based speech recognition system assumes that speech 30 is essentially either a quasi-periodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced sounds). It may be modeled as two sources—one a impulse train generator with pitch period P and a random noise generator which is controlled by a voice/unvoiced switch. 35 The output of the switch is then fed into a gain function estimated from the speech signal and scaled to feed a digital filter H(z) controlled by the vocal tract parameter characteristics of the speech being produced. All of the parameters for this model—the voiced/unvoiced switching, the pitch 40 period for voiced sounds, the gain parameter for the speech signal and the coefficient of the digital filter, vary slowly with time. In extracting the acoustic parameters from the user's speech input so that it can evaluated in light of a set vocal tract information from the excitation information. The cepstrum of a signal is computed by taking the Fourier (or similar) transform of the log spectrum. The principal advantage of extracting cepstral coefficients is that they are de-correlated and the diagonal covariances can be used with 50 HMMs. Since the human ear resolves frequencies nonlinearly across the audio spectrum, it has been shown that a front-end that operates in a similar non-linear way improves speech recognition performance.

Accordingly, instead of a typical linear prediction-based 55 analysis, the front-end of the NLQS speech recognition engine implements a simple, fast Fourier transform based filter bank designed to give approximately equal resolution on the Mel-scale. To implement this filter bank, a window of speech data (for a particular time frame) is transformed 60 using a software based Fourier transform and the magnitude taken. Each FFT magnitude is then multiplied by the corresponding filter gain and the results accumulated. The cepstral coefficients that are derived from this filter-bank analysis at the front end are calculated during a first partial 65 processing phase of the speech signal by using a Discrete Cosine Transform of the log filter bank amplitudes. These

16

cepstral coefficients are called Mel-Frequency Cepstral Coefficients (MFCC) and they represent some of the speech parameters transferred from the client side to characterize the acoustic features of the user's speech signal. These parameters are chosen for a number of reasons, including the fact that they can be quickly and consistently derived even across systems of disparate capabilities (i.e., for everything from a low power PDA to a high powered desktop system), they give good discrimination, they lend themselves to a number of useful recognition related manipulations, and they are relatively small and compact in size so that they can be transported rapidly across even a relatively narrow band link. Thus, these parameters represent the least amount of information that can be used by a subsequent server side system to adequately and quickly complete the recognition process.

To augment the speech parameters an energy term in the form of the logarithm of the signal energy is added. Accordingly, RMS energy is added to the 12 MFCC's to make 13 coefficients. These coefficients together make up the partially processed speech data transmitted in compressed form from the user's client system to the remote server side.

The performance of the present speech recognition system is enhanced significantly by computing and adding time derivatives to the basic static MFCC parameters at the server side. These two other sets of coefficients—the delta and acceleration coefficients representing change in each of the 13 values from frame to frame (actually measured across several frames), are computed during a second partial speech signal processing phase to complete the initial processing of the speech signal, and are added to the original set of coefficients after the latter are received. These MFCCs together with the delta and acceleration coefficients constitute the observation vector O, mentioned above that is used for determining the appropriate HMM for the speech data.

The delta and acceleration coefficients are computed using the following regression formula:

$$d_t = \Sigma_{\theta=1}^{\theta} [c_{t+\theta} - c_{t-\theta}] / 2\Sigma_{\theta=1}^{\theta} \theta^2$$

where d_t is a delta coefficient at time t computed in terms of the corresponding static coefficients:

$$d_t = [c_{t+\theta} - c_{t-\theta}]/2\theta$$

In a typical stand-alone implementation of a speech of HMMs, cepstral analysis is typically used to separate the 45 recognition system, the entire SR engine runs on a single client. In other words, both the first and second partial processing phases above are executed by the same DSP (or microprocessor) running a ROM or software code routine at the client's computing machine.

In contrast, because of several considerations, specifically-cost, technical performance, and client hardware uniformity, the present NLQS system uses a partitioned or distributed approach. While some processing occurs on the client side, the main speech recognition engine runs on a centrally located server or number of servers. More specifically, as noted earlier, capture of the speech signals, MFCC vector extraction and compression are implemented on the client's machine during a first partial processing phase. The routine is thus streamlined and simple enough to be implemented within a browser program (as a plug in module, or a downloadable applet for example) for maximum ease of use and utility. Accordingly, even very "thin" client platforms can be supported, which enables the use of the present system across a greater number of potential sites. The primary MFCCs are then transmitted to the server over the channel, which, for example, can include a dial-up INTER-NET connection, a LAN connection, a wireless connection

17

and the like. After decompression, the delta and acceleration coefficients are computed at the server to complete the initial speech processing phase, and the resulting observation vectors O_t are also determined.

General Aspects of Speech Recognition Engine

The speech recognition engine is also located on the server, and is based on a HTK-based recognition network compiled from a word-level network, a dictionary and a set of HMMs. The recognition network consists of a set of 10 nodes connected by arcs. Each node is either a HMM model instance or a word end. Each model node is itself a network consisting of states connected by arcs. Thus when fully compiled, a speech recognition network consists of HMM states connected by transitions. For an unknown input utter- 15 ance with T frames, every path from the start node to the exit node of the network passes through T HMM states. Each of these paths has log probability which is computed by summing the log probability of each individual transition in the path and the log probability of each emitting state 20 generating the corresponding observation. The function of the Viterbi decoder is find those paths through the network which have the highest log probability. This is found using the Token Passing algorithm. In a network that has many nodes, the computation time is reduced by only allowing 25 propagation of those tokens which will have some chance of becoming winners. This process is called pruning.

Natural Language Processor

In a typical natural language interface to a database, the user enters a question in his/her natural language, for example, English. The system parses it and translates it to a query language expression. The system then uses the query language expression to process the query and if the search is successful, a recordset representing the results is displayed in English either formatted as raw text or in a graphical form. For a natural language interface to work well involves a number of technical requirements.

For example, it needs to be robust—in the sentence 'What's the departments turnover' it needs to decide that the word whats—what's—what is. And it also has to determine that departments—department's. In addition to being robust, the natural language interface has to distinguish between the several possible forms of ambiguity that may exist in the natural language—lexical, structural, reference and ellipsis ambiguity. All of these requirements, in addition to the general ability to perform basic linguistic morphological operations of tokenization, tagging and grouping, are implemented within the present invention.

Tokenization is implemented by a text analyzer which 50 treats the text as a series of tokens or useful meaningful units that are larger than individual characters, but smaller than phrases and sentences. These include words, separable parts of words, and punctuation. Each token is associated with an offset and a length. The first phase of tokenization is the 55 process of segmentation which extracts the individual tokens from the input text and keeps track of the offset where each token originated in the input text. The tokenizer output lists the offset and category for each token. In the next phase of the text analysis, the tagger uses a built-in morphological 60 analyzer to look up each word/token in a phrase or sentence and internally lists all parts of speech. The output is the input string with each token tagged with a parts of speech notation. Finally the grouper which functions as a phrase extractor or phrase analyzer, determines which groups of words 65 form phrases. These three operations which are the foundations for any modern linguistic processing schemes, are fully

18

implemented in optimized algorithms for determining the single-best possible answer to the user's question.

SQL Database and Full-Text Query

Another key component of present system is a SQL-database. This database is used to store text, specifically the answer-question pairs are stored in full-text tables of the database. Additionally, the full-text search capability of the database allows full-text searches to be carried out.

While a large portion of all digitally stored information is in the form of unstructured data, primarily text, it is now possible to store this textual data in traditional database systems in character-based columns such as varchar and text. In order to effectively retrieve textual data from the database, techniques have to be implemented to issue queries against textual data and to retrieve the answers in a meaningful way where it provides the answers as in the case of the NLQS system.

There are two major types of textual searches: Property—This search technology first applies filters to documents in order to extract properties such as author, subject, type, word count, printed page count, and time last written, and then issues searches against those properties; Full-text—this search technology first creates indexes of all non-noise words in the documents, and then uses these indexes to support linguistic searches and proximity searches.

Two additional technologies are also implemented in this particular RDBMs: SQL Server also have been integrated: A Search service—a full-text indexing and search service that is called both index engine and search, and a parser that accepts full-text SQL extensions and maps them into a form that can be processed by the search engine.

The four major aspects involved in implementing full-text retrieval of plain-text data from a full-text-capable database are: Managing the definition of the tables and columns that are registered for full-text searches; Indexing the data in registered columns—the indexing process scans the character streams, determines the word boundaries (this is called word breaking), removes all noise words (this also is called stop words), and then populates a full-text index with the remaining words; Issuing queries against registered columns for populated full-text indexes; Ensuring that subsequent changes to the data in registered columns gets propagated to the index engine to keep the full-text indexes synchronized.

The underlying design principle for the indexing, querying, and synchronizing processes is the presence of a full-text unique key column (or single-column primary key) on all tables registered for full-text searches. The full-text index contains an entry for the non-noise words in each row together with the value of the key column for each row.

When processing a full-text search, the search engine returns to the database the key values of the rows that match the search criteria.

The full-text administration process starts by designating a table and its columns of interest for full-text search. Customized NLQS stored procedures are used first to register tables and columns as eligible for full-text search. After that, a separate request by means of a stored procedure is issued to populate the full-text indexes. The result is that the underlying index engine gets invoked and asynchronous index population begins. Full-text indexing tracks which significant words are used and where they are located. For example, a full-text index might indicate that the word "NLQS" is found at word number 423 and word number 982 in the Abstract column of the DevTools table for the row associated with a ProductID of 6. This index structure supports an efficient search for all items containing indexed

19

Case 3:08-cv-00863-MHP

words as well as advanced search operations, such as phrase searches and proximity searches. (An example of a phrase search is looking for "white elephant," where "white" is followed by "elephant". An example of a proximity search is looking for "big" and "house" where "big" occurs near 5 "house".) To prevent the full-text index from becoming bloated, noise words such as "a," "and," and "the" are ignored.

Extensions to the Transact-SQL language are used to construct full-text queries. The two key predicates that are 10 used in the NLQS are CONTAINS and FREETEXT.

The CONTAINS predicate is used to determine whether or not values in full-text registered columns contain certain words and phrases. Specifically, this predicate is used to search for:

A word or phrase.

The prefix of a word or phrase.

A word or phrase that is near another.

A word that is an inflectional form of another (for example, "drive" is the inflectional stem of "drives," 20 "drove," "driving," and "driven").

A set of words or phrases, each of which is assigned a different weighting.

The relational engine within SQL Server recognizes the CONTAINS and FREETEXT predicates and performs some 25 minimal syntax and semantic checking, such as ensuring that the column referenced in the predicate has been registered for full-text searches. During query execution, a full-text predicate and other relevant information are passed to the full-text search component. After further syntax and 30 semantic validation, the search engine is invoked and returns the set of unique key values identifying those rows in the table that satisfy the full-text search condition. In addition to the FREETEXT and CONTAINS, other predicates such as AND, LIKE, NEAR are combined to create the customized 35 NLQS SQL construct.

Full-Text Query Architecture of the SQL Database

The full-text query architecture is comprised of the following several components—Full-Text Query component, the SQL Server Relational Engine, the Full-Text provider and the Search Engine.

The Full-Text Query component of the SQL database accept a full-text predicate or rowset-valued function from the SQL Server; transform parts of the predicate into an internal format, and sends it to Search Service, which returns the matches in a rowset. The rowset is then sent back to SQL Server. SQL Server uses this information to create the resultset that is then returned to the submitter of the query.

The SQL Server Relational Engine accepts the CONTAINS and FREETEXT predicates as well as the CONTAINSTABLE() and FREETEXTTABLE() rowset-valued functions. During parse time, this code checks for conditions such as attempting to query a column that has not been registered for full-text search. If valid, then at run time, the ft_search_condition and context information is sent to the full-text provider. Eventually, the full-text provider returns a rowset to SQL Server, which is used in any joins (specified or implied) in the original query. The Full-Text Provider parses and validates the ft_search_condition, constructs the appropriate internal representation of the full-text search condition, and then passes it to the search engine. The result is returned to the relational engine by means of a rowset of rows that satisfy ft_search_condition.

Client Side System 150

The architecture of client-side system 150 of Natural Language Query System 100 is illustrated in greater detail in

20

FIG. 2. Referring to FIG. 2, the three main processes effectuated by Client System 150 are illustrated as follows: Initialization process 200A consisting of SRE 201, Communication 202 and Microsoft (MS) Agent 203 routines; an iterative process 200B consisting of two sub-routines: a) Receive User Speech 208-made up of SRE 204 and Communication 205; and b) Receive Answer from Server 207—made up of MS Speak Agent 206, Communication 209, Voice data file 210 and Text to Speech Engine 211. Finally, un-initialization process 200C is made up of three sub-routines: SRE 212, Communication 213, and MS Agent 214. Each of the above three processes are described in detail in the following paragraphs. It will be appreciated by those skilled in the art that the particular implementation for 15 such processes and routines will vary from client platform to platform, so that in some environments such processes may be embodied in hard-coded routines executed by a dedicated DSP, while in others they may be embodied as software routines executed by a shared host processor, and in still others a combination of the two may be used.

Initialization at Client System 150

The initialization of the Client System 150 is illustrated in FIG. 2—2 and is comprised generally of 3 separate initializing processes: client-side Speech Recognition Engine 220A, MS Agent 220B and Communication processes 220C

Initialization of Speech Recognition Engine 220A

Speech Recognition Engine 155 is initialized and configured using the routines shown in 220A. First, an SRE COM Library is initialized. Next, memory 220 is allocated to hold Source and Coder objects, are created by a routine 221. Loading of configuration file 221A from configuration data file 221B also takes place at the same time that the SRE Library is initialized. In configuration file **221**B, the type of the input of Coder and the type of the output of the Coder are declared. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches. Accordingly, they are not discussed in detail herein. Next, Speech and Silence components of an utterance are calibrated using a routine 222, in a procedure that is also well-known in the art. To calibrate the speech and silence components, the user preferably articulates a sentence that is displayed in a text box on the screen. The SRE library then estimates the noise and other parameters required to find e silence and speech elements of future user utterances.

Initialization of MS Agent 220B

The software code used to initialize and set up a MS Agent 220B is also illustrated in FIG. 2—2. The MS Agent 220B routine is responsible for coordinating and handling the actions of the animated agent 157 (FIG. 1). This initialization thus consists of the following steps:

- 1. Initialize COM library 223. This part of the code initializes the COM library, which is required to use ActiveX Controls, which controls are well-known in the art.
- 2. Create instance of Agent Server 224—this part of the code creates an instance of Agent ActiveX control.
- 3. Loading of MS Agent 225—this part of the code loads MS Agent character from a specified file 225A containing general parameter data for the Agent Character, such as the overall appearance, shape, size, etc.
- Get Character Interface 226—this part of the code gets an appropriate interface for the specified character; for

example, characters may have different control/interaction capabilities that can be presented to the user.

21

- 5. Add Commands to Agent Character Option 227—this part of the code adds commands to an Agent Properties sheet, which sheet can be accessed by clicking on the icon that appears in the system tray, when the Agent character is loaded e.g., that the character can Speak, how he/she moves, TTS Properties, etc.
- Show the Agent Character 228—this part of the code displays the Agent character on the screen so it can be 10 seen by the user;
- AgentNotifySink—to handle events. This part of the code creates AgentNotifySink object 229, registers it at 230 and then gets the Agent Properties interface 231.
 The property sheet for the Agent character is assigned 15 using routine 232.
- Do Character Animations 233—This part of the code plays specified character animations to welcome the user to NLQS 100.

The above then constitutes the entire sequence required to 20 initialize the MS Agent. As with the SRE routines, the MS Agent routines can be implemented in any suitable and conventional fashion by those skilled in the art based on the present teachings. The particular structure, operation, etc. of such routines is not critical, and thus they are not discussed 25 in detail herein.

In a preferred embodiment, the MS Agent is configured to have an appearance and capabilities that are appropriate for the particular application. For instance, in a remote learning application, the agent has the visual form and mannerisms/ attitude/gestures of a college professor. Other visual props (blackboard, textbook, etc.) may be used by the agent and presented to the user to bring to mind the experience of being in an actual educational environment. The characteristics of the agent may be configured at the client side 150, 35 and/or as part of code executed by a browser program (not shown) in response to configuration data and commands from a particular web page. For example, a particular website offering medical services may prefer to use a visual image of a doctor. These and many other variations will be 40 apparent to those skilled in the art for enhancing the humanlike, real-time dialog experience for users.

Initialization of Communication Link 160A

The initialization of Communication Link 160A is shown with reference to process 220C FIG. 2—2. Referring to FIG. 2—2, this initialization consists of the following code components: Open INTERNET Connection 234—this part of the code opens an INTERNET Connection and sets the parameter for the connection. Then Set Callback Status routine 235 sets the callback status so as to inform the user of the status of connection. Finally Start New HTTP INTERNET Session 236 starts a new INTERNET session. The details of Communications Link 160 and the set up process 220C are not critical, and will vary from platform to platform. Again, in some cases, users may use a low-speed dial-up connection, a dedicated high speed switched connection (T1 for example), an always-on xDSL connection, a wireless connection, and the like.

Iterative Processing of Queries/Answers

As illustrated in FIG. 3, once initialization is complete, an iterative query/answer process is launched when the user presses the Start Button to initiate a query. Referring to FIG. 3, the iterative query/answer process consists of two main sub-processes implemented as routines on the client side 65 system 150: Receive User Speech 240 and Receive User Answer 243. The Receive User Speech 240 routine receives

22

speech from the user (or another audio input source), while the Receive User Answer 243 routine receives an answer to the user's question in the form of text from the server so that it can be converted to speech for the user by text-to-speech engine 159. As used herein, the term "query" is referred to in the broadest sense to refer, to either a question, a command, or some form of input used as a control variable by the system. For example, a query may consist of a question directed to a particular topic, such as "what is a network" in the context of a remote learning application. In an e-commerce application a query might consist of a command to "list all books by Mark Twain" for example. Similarly, while the answer in a remote learning application consists of text that is tendered into audible form by the text to speech engine 159, it could also be returned as another form of multi-media information, such as a graphic image, a sound file, a video file, etc. depending on the requirements of the particular application. Again, given the present teachings concerning the necessary structure, operation, functions, performance, etc., of the client-side Receive User Speech 240 and Receiver User Answer 243 routines, one of ordinary skill in the art could implement such in a variety of ways.

Receive User Speech—As illustrated in FIG. 3, the Receive User Speech routine 240 consists of a SRE 241 and a Communication 242 process, both implemented again as routines on the client side system 150 for receiving and partially processing the user's utterance. SRE routine 241 uses a coder 248 which is prepared so that a coder object receives speech data from a source object. Next the Start Source 249 routine is initiated. This part of the code initiates data retrieval using the source Object which will in turn be given to the Coder object. Next, MFCC vectors 250 are extracted from the Speech utterance continuously until silence is detected. As alluded to earlier, this represents the first phase of processing of the input speech signal, and in a preferred embodiment, it is intentionally restricted to merely computing the MFCC vectors for the reasons already expressed above. These vectors include the 12 cepstral coefficients and the RMS energy term, for a total of 13 separate numerical values for the partially processed speech signal.

In some environments, nonetheless, it is conceivable that the MFCC delta parameters and MFCC acceleration parameters can also be computed at client side system 150, depending on the computation resources available, the transmission bandwidth in data link 160A available to server side system 180, the speed of a transceiver used for carrying data in the data link, etc. These parameters can be determined automatically by client side system upon initializing SRE 155 (using some type of calibration routine to measure resources), or by direct user control, so that the partitioning of signal processing responsibilities can be optimized on a case-by-case basis. In some applications, too, server side system 180 may lack the appropriate resources or routines for completing the processing of the speech input signal. Therefore, for some applications, the allocation of signal processing responsibilities may be partitioned differently, to the point where in fact both phases of the speech signal processing may take place at client side system 150 so that the speech signal is completely—rather than partiallyprocessed and transmitted for conversion into a query at server side system 180.

Again in a preferred embodiment, to ensure reasonable accuracy and real-time performance from a query/response perspective, sufficient resources are made available in a client side system so that 100 frames per second of speech data can be partially processed and transmitted through link

160A. Since the least amount of information that is necessary to complete the speech recognition process (only 13 coefficients) is sent, the system achieves a real-time performance that is believed to be highly optimized, because other latencies (i.e., client-side computational latencies, packet 5 formation latencies, transmission latencies) are minimized. It will be apparent that the principles of the present invention can be extended to other SR applications where some other methodology is used for breaking down the speech input signal by an SRE (i.e., non-MFCC based). The only criteria 10 is that the SR processing be similarly dividable into multiple phases, and with the responsibility for different phases being handled on opposite sides of link 160A depending on overall system performance goals, requirements and the like. This functionality of the present invention can thus be achieved 15 on a system-by-system basis, with an expected and typical amount of optimization being necessary for each particular implementation.

Thus, the present invention achieves a response rate performance that is tailored in accordance with the amount 20 of information that is computed, coded and transmitted by the client side system **150**. So in applications where real-time performance is most critical, the least possible amount of extracted speech data is transmitted to reduce these latencies, and, in other applications, the amount of extracted 25 speech data that is processed, coded and transmitted can be varied

Communication—transmit communication module **242** is used to implement the transport of data from the client to the server over the data link **160**A, which in a preferred embodiment is the INTERNET. As explained above, the data consists of encoded MFCC vectors that will be used at then server-side of the Speech Recognition engine to complete the speech recognition decoding. The sequence of the communication is as follows:

OpenHTTPRequest **251**—this part of the code first converts MFCC vectors to a stream of bytes, and then processes the bytes so that it is compatible with a protocol known as HTTP. This protocol is well-known in the art, and it is apparent that for other data links another suitable protocol 40 would be used.

- 1. Encode MFCC Byte Stream **251**—this part of the code encodes the MFCC vectors, so that they can be sent to the server via HTTP.
- Send data 252—this part of the code sends MFCC 45
 vectors to the server using the INTERNET connection
 and the HTTP protocol.

Wait for the Server Response 253—this part of the code monitors the data link 160A a response from server side system 180 arrives. In summary, the MFCC parameters are 50 extracted or observed on-the-fly from the input speech signal. They are then encoded to a HTTP byte stream and sent in a streaming fashion to the server before the silence is detected—i.e. sent to server side system 180 before the utterance is complete. This aspect of the invention also 55 facilitates a real-time behavior, since data can be transmitted and processed even while the user is still speaking.

Receive Answer from Server 243 is comprised of the following modules as shown in FIG. 3.: MS Agent 244, Text-to-Speech Engine 245 and receive communication 60 modules 246. All three modules interact to receive the answer from server side system 180. As illustrated in FIG. 3, the receive communication process consists of three separate processes implemented as a receive routine on client side system 150: a Receive the Best Answer 258 65 receives the best answer over data link 160B (the HTTP communication channel). The answer is de-compressed at

24

259 and then the answer is passed by code 260 to the MS Agent 244, where it is received by code portion 254. A routine 255 then articulates the answer using text-to-speech engine 257. Of course, the text can also be displayed for additional feedback purposes on a monitor used with client side system 150. The text to speech engine uses a natural language voice data file 256 associated with it that is appropriate for the particular language application (i.e., English, French, German, Japanese, etc.). As explained earlier when the answer is something more than text, it can be treated as desired to provide responsive information to the user, such as with a graphics image, a sound, a video clip, etc.

Uninitialization

The un-initialization routines and processes are illustrated in FIG. 4. Three functional modules are used for uninitializing the primary components of the client side system 150; these include SRE 270, Communications 271 and MS Agent 272 un-initializing routines. To un-initialize SRE 220A, memory that was allocated in the initialization phase is de-allocated by code 273 and objects created during such initialization phase are deleted by code 274. Similarly, as illustrated in FIG. 4, to un-initialize Communications module 220C the INTERNET connection previously established with the server is closed by code portion 275 of the Communication Un-initialization routine 271. Next the INTER-NET session created at the time of initialization is also closed by routine 276. For the un-initialization of the MS Agent 220B, as illustrated in FIG. 4, MS Agent Un-initialization routine 272 first releases the Commands Interface 227 using routine 277. This releases the commands added to the property sheet during loading of the agent character by routine 225. Next the Character Interface initialized by routine 226 is released by routine 278 and the Agent is unloaded at 279. The Sink Object Interface is then also released 280 followed by the release of the Property Sheet Interface 281. The Agent Notify Sink 282 then un-registers the Agent and finally the Agent Interface 283 is released which releases all the resources allocated during initialization steps identified in FIG. 2—2.

It will be appreciated by those skilled in the art that the particular implementation for such un-initialization processes and routines in FIG. 4 will vary from client platform to client platform, as for the other routines discussed above. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches without undue effort. Accordingly, they are not discussed in detail herein.

Description of Server Side System 180

Introduction

A high level flow diagram of the set of preferred processes implemented on server side system 180 of Natural Language Query System 100 is illustrated in FIGS. 11A through FIG. 11C. In a preferred embodiment, this process consists of a two step algorithm for completing the processing of the speech input signal, recognizing the meaning of the user's query, and retrieving an appropriate answer/response for such query.

The 1st step as illustrated in FIG. 11A can be considered a high-speed first-cut pruning mechanism, and includes the following operations: after completing processing of the speech input signal, the user's query is recognized at step 1101, so that the text of the query is simultaneously sent to Natural Language Engine 190 (FIG. 1) at step 1107, and to DB Engine 186 (also FIG. 1) at step 1102. By "recognized"

in this context it is meant that the user's query is converted into a text string of distinct native language words through the HMM technique discussed earlier.

At NLE **190**, the text string undergoes morphological linguistic processing at step **1108**: the string is tokenized the tags are tagged and the tagged tokens are grouped Next the noun phrases (NP) of the string are stored at **1109**, and also copied and transferred for use by DB Engine **186** during a DB Process at step **1110**. As illustrated in FIG. **11A**, the string corresponding to the user's query which was sent to the DB Engine **186** at **1102**, is used together with the NP received from NLE **190** to construct an SQL Query at step **1103**. Next, the SQL query is executed at step **1104**, and a record set of potential questions corresponding to the user's query are received as a result of a full-text search at **1105**, which are then sent back to NLE **190** in the form of an array at step **1106**.

As can be seen from the above, this first step on the server side processing acts as an efficient and fast pruning mechanism so that the universe of potential "hits" corresponding to the user's actual query is narrowed down very quickly to a manageable set of likely candidates in a very short period of time

Referring to FIG. 11B, in contrast to the first step above, 25 the 2^{nd} step can be considered as the more precise selection portion of the recognition process. It begins with linguistic processing of each of the stored questions in the array returned by the full-text search process as possible candidates representing the user's query. Processing of these 30 stored questions continues in NLE 190 as follows: each question in the array of questions corresponding to the record set returned by the SQL full-text search undergoes morphological linguistic processing at step 1111: in this operation, a text string corresponding to the retrieved candidate question is tokenized, the tags are tagged and the tagged tokens are grouped. Next, noun phrases of the string are computed and stored at step 1112. This process continues iteratively at point 1113, and the sequence of steps at 1118, 1111, 1112, 1113 are repeated so that an NP for each 40 retrieved candidate question is computed and stored. Once an NP is computed for each of the retrieved candidate questions of the array, a comparison is made between each such retrieved candidate question and the user's query based on the magnitude of the NP value at step 1114. This process 45 is also iterative in that steps 1114, 1115, 1116, 1119 are repeated so that the comparison of the NP for each retrieved candidate question with that of the NP of the user's query is completed. When there are no more stored questions in the array to be processed at step 1117, the stored question that 50 has the maximum NP relative to the user's query, is identified at 1117A as the stored question which best matches the user's query.

Notably, it can be seen that the second step of the recognition process is much more computationally intensive 55 than the first step above, because several text strings are tokenized, and a comparison is made of several NPs. This would not be practical, nonetheless, if it were not for the fact that the first step has already quickly and efficiently reduced the candidates to be evaluated to a significant degree. Thus, 60 this more computationally intensive aspect of the present invention is extremely valuable, however because it yields extremely high accuracy in the overall query recognition process. In this regard, therefore, this second step of the query recognition helps to ensure the overall accuracy of the 65 system, while the first step helps to maintain a satisfactory speed that provides a real-time feel for the user.

26

As illustrated in FIG. 11C, the last part of the query/response process occurs by providing an appropriate matching answer/response to the user. Thus, an identity of a matching stored question is completed at step 1120. Next a file path corresponding to an answer of the identified matching question is extracted at step 1121. Processing continues so that the answer is extracted from the file path at 1122 and finally the answer is compressed and sent to client side system 150 at step 1123.

The discussion above is intended to convey a general overview of the primary components, operations, functions and characteristics of those portions of NLQS system 100 that reside on server side system 180. The discussion that follows describes in more detail the respective sub-systems.

Software Modules used in Server Side System 180

The key software modules used on server-side system 180 of the NLQS system are illustrated in FIG. 5. These include generally the following components: a Communication module 500—identified as CommunicationServer ISAPI 500A (which is executed by SRE Server-side 182-FIG. 1 and is explained in more detail below), and a database process DBProcess module 501 (executed by DB Engine **186**—FIG. 1). Natural language engine module **500**C (executed by NLE 190-FIG. 1) and an interface 500B between the NLE process module 500C and the DBProcess module 500B. As shown here, CommunicationServerISAPI 500A includes a server-side speech recognition engine and appropriate communication interfaces required between client side system 150 and server side system 180. As further illustrated in FIG. 5, server-side logic of Natural Language Query System 100 also can be characterized as including two dynamic link library components: CommunicationServerISAPI 500 and DBProcess 501. The CommunicationServerIASPI 500 is comprised of 3 sub-modules: Server-side Speech Recognition Engine module 500A; Interface module 500B between Natural Language Engine modules 500C and DBProcess 501; and the Natural Language Engine modules

DB Process **501** is a module whose primary function is to connect to a SQL database and to execute an SQL query that is composed in response to the user's query. In addition, this module interfaces with logic that fetches the correct answer from a file path once this answer is passed to it from the Natural Language Engine module **500**C.

Speech Recognition Sub-System 182 on Server-Side System 180

The server side speech recognition engine module 500A is a set of distributed components that perform the necessary functions and operations of speech recognition engine 182 (FIG. 1) at server-side 180. These components can be implemented as software routines that are executed by server side 180 in conventional fashion. Referring to FIG. 4A, a more detailed break out of the operation of the speech recognition components 600 at the server-side can be seen as follows:

Within a portion 601 of the server side SRE module 500A, the binary MFCC vector byte stream corresponding to the speech signal's acoustic features extracted at client side system 150 and sent over the communication channel 160 is received. The MFCC acoustic vectors are decoded from the encoded HTTP byte stream as follows: Since the MFCC vectors contain embedded NULL characters, they cannot be transferred in this form to server side system 180 as such using HTTP protocol. Thus the MFCC vectors are first encoded at client-side 150 before transmission in such a way that all the speech data is converted into a stream of bytes

without embedded NULL characters in the data. At the very end of the byte stream a single NULL character is introduced to indicate the termination of the stream of bytes to be transferred to the server over the INTERNET 160A using HTTP protocol.

As explained earlier, to conserve latency time between the client and server, a smaller number of bytes (just the 13 MFCC coefficients) are sent from client side system 150 to server side system 180. This is done automatically for each platform to ensure uniformity, or can be tailored by the particular application environment—i.e., such as where it is determined that it will take less time to compute the delta and acceleration coefficients at the server (26 mote calculations), than it would take to encode them at the client, transmit them, and then decode them from the HTTP stream. In general, since server side system 180 is usually better equipped to calculate the MFCC delta and acceleration parameters, this is a preferable choice. Furthermore, there is generally more control over server resources compared to the client's resources, which means that future upgrades, optimizations, etc., can be disseminated and shared by all to make overall system performance more reliable and predictable. So, the present invention can accommodate even the worst-case scenario where the client's machine may be quite thin and may just have enough resources to capture the speech input data and do minimal processing.

Dictionary Preparation & Grammar Files

Referring to FIG. 4A, within code block 605, various options selected by the user (or gleaned from the user's status within a particular application) are received. For instance, in the case of a preferred remote learning system, Course, Chapter and/or Section data items are communicated. In the case of other applications (such as e-commerce) other data options are communicated, such as the Product Class, Product Category, Product Brand, etc. loaded for viewing within his/her browser. These selected options are based on the context experienced by the user during an interactive process, and thus help to limit and define the scope—i.e. grammars and dictionaries that will be dynamically loaded to speech recognition engine 182 (FIG. 1) for Viterbi decoding during processing of the user speech utterance. For speech recognition to be optimized both grammar and dictionary files are used in a preferred embodiment. A $_{45}$ Grammar file supplies the universe of available user queries; i.e., all the possible words that are to be recognized. The Dictionary file provides phonemes (the information of how a word is pronounced—this depends on the specific native language files that are installed—for example, UK English or US English) of each word contained in the grammar file. It is apparent that if all the sentences for a given environment that can be recognized were contained in a single grammar file then recognition accuracy would be deteriorated and the loading time alone for such grammar and dictionary files would impair the speed of the speech recognition process.

To avoid these problems, specific grammars are dynamically loaded or actively configured as the current grammar according to the user's context, i.e., as in the case of a remote learning system, the Course, Chapter and/or Section 60 Database Processor 186 Operation—DBProcess selected. Thus the grammar and dictionary files are loaded dynamically according to the given Course, Chapter and/or Section as dictated by the user, or as determined automatically by an application program executed by the user.

The second code block 602 implements the initialization 65 of Speech Recognition engine 182 (FIG. 1). The MFCC vectors received from client side system 150 along with the

grammar filename and the dictionary file names are introduced to this block to initialize the speech decoder.

As illustrated in FIG. 4A, the initialization process 602 uses the following sub-routines: A routine 602a for loading an SRE library. This then allows the creation of an object identified as External Source with code 602b using the received MFCC vectors. Code 602c allocates memory to hold the recognition objects. Routine 602d then also creates and initializes objects that are required for the recognition such as: Source, Coder, Recognizer and Results Loading of the Dictionary created by code 602e, Hidden Markov Models (HMMs) generated with code 602f; and Loading of the Grammar file generated by routine 602g.

Speech Recognition 603 is the next routine invoked as illustrated in FIG. 4A, and is generally responsible for completing the processing of the user speech signals input on the client side 150, which, as mentioned above, are preferably only partially processed (i.e., only MFCC vectors are computed during the first phase) when they are transmitted across link 160. Using the functions created in External Source by subroutine 602b, this code reads MFCC vectors, one at a time from an External Source 603a, and processes them in block 603b to realize the words in the speech pattern that are symbolized by the MFCC vectors captured at the client. During this second phase, an additional 13 delta coefficients and an additional 13 acceleration coefficients are computed as part of the recognition process to obtain a total of 39 observation vectors O, referred to earlier. Then, using a set of previously defined Hidden Markov Models (HMMs), the words corresponding to the user's speech utterance are determined in the manner described earlier. This completes the word "recognition" aspect of the query processing, which results are used further below to complete the query processing operations.

It will be appreciated by those skilled in the art that the distributed nature and rapid performance of the word recognition process, by itself, is extremely useful and may be implemented in connection with other environments that do not implicate or require additional query processing operations. For example, some applications may simply use individual recognized words for filling in data items on a computer generated form, and the aforementioned systems and processes can provide a rapid, reliable mechanism for doing so.

Once the user's speech is recognized, the flow of SRE 182 passes to Un-initialize SRE routine 604 where the speech engine is un-initialized as illustrated. In this block all the objects created in the initialization block are deleted by routine 604a, and memory allocated in the initialization block during the initialization phase are removed by routine

Again, it should be emphasized that the above are merely illustrative of embodiments for implementing the particular routines used on a server side speech recognition system of the present invention. Other variations of the same that achieve the desired functionality and objectives of the present invention will be apparent from the present teach-

Construction of an SQL Query used as part of the user query processing is illustrated in FIG. 4B, a SELECT SQL statement is preferably constructed using a conventional CONTAINS predicate. Module 950 constructs the SQL query based on this SELECT SQL statement, which query is used for retrieving the best suitable question stored in the database corresponding to the user's articulated query, (desDocument 52-4

29

ignated as Question here). A routine 951 then concatenates a table name with the constructed SELECT statement. Next, the number of words present in each Noun Phrase of Question asked by the user is calculated by routine 952. Then memory is allocated by routine 953 as needed to 5 accommodate all the words present in the NP. Next a word List (identifying all the distinct words present in the NP) is obtained by routine 954. After this, this set of distinct words are concatenated by routine 955 to the SQL Query separated with a NEAR () keyword. Next, the AND keyword is concatenated to the SQL Query by routine 956 after each NP. Finally memory resources are freed by code 957 so as to allocate memory to store the words received from NP for any next iteration. Thus, at the end of this process, a completed SQL Query corresponding to the user's articu- 15 lated question is generated.

Connection to SQL Server—As illustrated in FIG. 4C, after the SQL Query is constructed by routine 710, a routine 711 implements a connection to the query database 717 to continue processing of the user query. The connection 20 sequence and the subsequent retrieved record set is implemented using routines 700 which include the following:

- 1. Server and database names are assigned by routine **711**A to a DBProcess member variable
- 2. A connection string is established by routine **711**B;
- 3. The SQL Server database is connected under control of code **711**C
- 4. The SQL Query is received by routine 712A
- 5. The SQL Query is executed by code **712**B
- 6. Extract the total number of records retrieved by the 30 query-713
- 7. Allocate the memory to store the total number of paired questions—713
- 8. Store the entire number of paired questions into an array-713

Once the Best Answer ID is received at 716 FIG. 4C, from the NLE 14 (FIG. 5), the code corresponding 716C receives it passes it to code in 716B where the path of the Answer file is determined using the record number. Then the file is opened 716C using the path passed to it and the contents of 40 the file corresponding to the answer is read. Then the answer is compressed by code in 716D and prepared for transmission over the communication channel 160B (FIG. 1).

NLQS Database 188—Table Organization

FIG. 6 illustrates a preferred embodiment of a logical structure of tables used in a typical NLQS database 188 (FIG. 1). When NLOS database 188 is used as part of NLOS query system 100 implemented as a remote learning/training environment, this database will include an organizational 50 multi-level hierarchy that consists typically of a Course 701, which is made of several chapters 702, 703, 704. Each of these chapters can have one or more Sections 705, 706, 707 as shown for Chapter 1. A similar structure can exist for Chapter 2, Chapter 3 . . . Chapter N. Each section has a set 55 of one or more question—answer pairs 708 stored in tables described in more detail below. While this is an appropriate and preferable arrangement for a training/learning application, it is apparent that other implementations would be possible and perhaps more suitable for other applications 60 such as e-commerce, e-support, INTERNET browsing, etc., depending on overall system parameters.

It can be seen that the NLQS database 188 organization is intricately linked to the switched grammar architecture described earlier. In other words, the context (or environ- 65 ment) experienced by the user can be determined at any moment in time based at the selection made at the section

30

level, so that only a limited subset of question-answer pairs 708 for example are appropriate for section 705. This in turn means that only a particular appropriate grammar for such question-answer pairs may be switched in for handling user queries while the user is experiencing such context. In a similar fashion, an e-commerce application for an INTER-NET based business may consist of a hierarchy that includes a first level "home" page 701 identifying user selectable options (product types, services, contact information, etc.), a second level may include one or more "product types" pages 702, 703, 704, a third page may include particular product models 705, 706, 707, etc., and with appropriate question-answer pairs 708 and grammars customized for handling queries for such product models. Again, the particular implementation will vary from application to application, depending on the needs and desires of such business, and a typical amount of routine optimization will be necessary for each such application.

Table Organization

In a preferred embodiment, an independent database is used for each Course. Each database in turn can include three types of tables as follows: a Master Table as illustrated in FIG. 7A, at least one Chapter Table as illustrated in FIG. 7B and at least one Section Table as illustrated in FIG. 7C.

As illustrated in FIG. 7A, a preferred embodiment of a Master Table has six columns—Field Name 701A, Data Type 702A, Size 703A, Null 704A, Primary Key 705A and Indexed 706A. These parameters are well-known in the art of database design and structure. The Master Table has only two fields—Chapter Name 707A and Section Name 708A. Both ChapterName and Section Name are commonly indexed.

A preferred embodiment of a Chapter Table is illustrated in FIG. 7B. As with the Master Table, the Chapter Table has six (6) columns—Field Name 720, Data Type 721, Size 722, Null 723, Primary Key 724 and Indexed 725. There are nine (9) rows of data however, in this case,—Chapter_ID 726, Answer_ID 727, Section Name 728, Answer_Title 729, PairedQuestion 730, AnswerPath 731, Creator 732, Date of Creation 733 and Date of Modification 734.

An explanation of the Chapter Table fields is provided in FIG. 7C. Each of the eight (8) Fields 720 has a description 735 and stores data corresponding to:

AnswerID 727—an integer that is automatically incremented for each answer given for user convenience

Section_Name 728—the name of the section to which the particular record belongs. This field along with the AnswerID is used as the primary key

Answer_Title 729—A short description of the title of the answer to the user query

PairedQuestion 730—Contains one or more combinations of questions for the related answers whose path is stored in the next column AnswerPath

AnswerPath 731—contains the path of a file, which contains the answer to the related questions stored in the previous column; in the case of a pure question/ answer application, this file is a text file, but, as mentioned above, could be a multi-media file of any kind transportable over the data link 160

Creator 732—Name of Content Creator

Date_of_Creation 733—Date on which content was created

Date of Modification 734—Date on which content was changed or modified

A preferred embodiment of a Section Table is illustrated in FIG. 7D. The Section Table has six (6) columns—Field

Name 740, Data Type 741, Size 742, Null 743, Primary Key 744 and Indexed 745. There are seven (7) rows of data-Answer ID 746, Answer Title 747, PairedQuestion 748, AnswerPath 749, Creator 750, Date of Creation 751 and Date of Modification 752. These names correspond to the 5 same fields, columns already described above for the Master Table and Chapter Table.

Again, this is a preferred approach for the specific type of learning/training application described herein. Since the number of potential applications for the present invention is 10 quite large, and each application can be customized, it is expected that other applications (including other learning/ training applications) will require and/or be better accommodated by another table, column, and field structure/ hierarchy.

Search Service and Search Engine-A query text search service is performed by an SQL Search System 1000 shown in FIG. 10. This system provides querying support to process full-text searches. This is where full-text indexes reside.

In general, SQL Search System determines which entries in a database index meet selection criteria specified by a particular text query that is constructed in accordance with an articulated user speech utterance. The Index Engine 1011B is the entity that populates the Full-Text Index tables with indexes which correspond to the indexable units of text for the stored questions and corresponding answers. It scans through character strings, determines word boundaries, removes all noise words and then populates the full-text index with the remaining words. For each entry in the full text database that meets the selection criteria, a unique key column value and a ranking value are returned as well. Catalog set 1013 is a file-system directory that is accessible only by an Administrator and Search Service 1010. Full-text indexes 1014 are organized into full-text catalogs, which are referenced by easy to handle names. Typically, full-text index data for an entire database is placed into a single full-text catalog.

The schema for the full-text database as described (FIG. 7, FIG. 7A, FIG. 7B, FIG. 7C, FIG. 7D) is stored in the tables 1006 shown in FIG. 10. Take for example, the tables required to describe the structure the stored question/answer pairs required for a particular course. For each table-Course Table, Chapter Table, Section Table, there are fields—column information that define each parameters that make up the logical structure of the table. This information is stored in User and System tables 1006. The key values corresponding to those tables are stored as Full-Text catalogs 1013. So when processing a full-text search, the search engine returns to the SQL Server the key values of the rows that match the search criteria. The relational engine then uses this information to respond to the query.

As illustrated in FIG. 10, a Full-Text Query Process is implemented as follows:

- 1. A guery 1001 that uses a SQL full-text construct 55 generated by DB processor 186 is submitted to SQL Relational Engine 1002.
- 2. Queries containing either a CONTAINS or FREETEXT predicate are rewritten by routine 1003 so that a responsive rowset returned later from Full-Text Provider 1007 60 will be automatically joined to the table that the predicate is acting upon. This rewrite is a mechanism used to ensure that these predicates are a seamless extension to a traditional SQL Server. After the compiled query is internally rewritten and checked for correctness in item 65 1003, the query is passed to RUN TIME module 1004. The function of module 1004 is to convert the rewritten

32

SQL construct to a validated run-time process before it is sent to the Full-Text Provider, 1007.

- 3. After this, Full-Text Provider 1007 is invoked, passing the following information for the query:
- a. A ft_search_condition parameter (this is a logical flag indicating a full text search condition)
- b. A name of a full-text catalog where a full-text index of a table resides
- c. A locale ID to be used for language (for example, word breaking)
- d. Identities of a database, table, and column to be used in the query
- e. If the query is comprised of more than one full-text construct; when this is the case Full-text provider 1007 is invoked separately for each construct.
- 4. SQL Relational Engine 1002 does not examine the contents of ft_search_condition. Instead, this information is passed along to Full-text provider 1007, which verifies the validity of the query and then creates an appropriate internal representation of the full-text search condition.
- 5. The query request/command 1008 is then passed to Querying Support 1011A.
- 6. Querying Support 1012 returns a rowset 1009 from Full-Text Catalog 1013 that contains unique key column values for any rows that match the full-text search criteria. A rank value also is returned for each row.
- 7. The rowset of key column values 1009 is passed to SQL Relational Engine 1002. If processing of the query implicates either a CONTAINSTABLE() or FREET-EXTTABLE() function, RANK values are returned; otherwise, any rank value is filtered out.
- 8. The rowset values 1009 are plugged into the initial query with values obtained from relational database 1006, and a result set 1015 is then returned for further processing to yield a response to the user.

At this stage of the query recognition process, the speech utterance by the user has already been rapidly converted into a carefully crafted text query, and this text query has been initially processed so that an initial matching set of results can be further evaluated for a final determination of the appropriate matching question/answer pair. The underlying principle that makes this possible is the presence of a full-text unique key column for each table that is registered for full-text searches. Thus when processing a full-text search, SQL Search Service 1010 returns to SQL server 1002 the key values of the rows that match the database. In maintaining these full-text databases 1013 and full text indexes 1014, the present invention has the unique characteristic that the full-text indices 1014 are not updated instantly when the full-text registered columns are updated. This operation is eliminated, again, to reduce recognition latency, increase response speed, etc. Thus, as compared to other database architectures, this updating of the full-text index tables, which would otherwise take a significant time, is instead done asynchronously at a more convenient time.

Interface Between NLE 190 and DB Processor 188

The result set 1015 of candidate questions corresponding to the user query utterance are presented to NLE 190 for further processing as shown in FIG. 4D to determine a "best" matching question/answer pair. An NLE/DBProcessor interface module coordinates the handling of user queries, analysis of noun-phrases (NPs) of retrieved questions sets from the SQL query based on the user query, comparing the retrieved question NPs with the user query NP, etc. between NLE 190 and DB Processor 188. So, this part of the

33

server side code contains functions, which interface processes resident in both NLE block 190 and DB Processor block 188. The functions are illustrated in FIG. 4D; As seen here, code routine 880 implements functions to extract the Noun Phrase (NP) list from the user's question. This part of 5 the code interacts with NLE 190 and gets the list of Noun Phrases in a sentence articulated by the user. Similarly, Routine 813 retrieves an NP list from the list of corresponding candidate/paired questions 1015 and stores these questions into an (ranked by NP value) array. Thus, at this point, 10 NP data has been generated for the user query, as well as for the candidate questions 1015. As an example of determining the noun phrases of a sentence such as: "What issues have guided the President in considering the impact of foreign trade policy on American businesses?" NLE 190 would 15 return the following as noun phrases: President, issues, impact of foreign trade policy, American businesses, impact, impact of foreign trade, foreign trade policy, trade, trade policy, policy, businesses. The methodology used by NLE 190 will thus be apparent to those skilled in the 20 art from this set of noun phrases and noun sub-phrases generated in response to the example query.

Next, a function identified as Get Best Answer ID **815** is implemented. This part of the code gets a best answer ID corresponding to the user's query. To do this, routines **813**A, 25 **813**B first find out the number of Noun phrases for each entry in the retrieved set **1015** that match with the Noun phrases in the user's query. Then routine **815***a* selects a final result record from the candidate retrieved set **1015** that contains the maximum number of matching Noun phrases. 30

Conventionally, nouns are commonly thought of as "naming" words, and specifically as the names of "people, places, or things". Nouns such as John, London, and computer certainly fit this description, but the types of words classified by the present invention as nouns is much broader than this. 35 Nouns can also denote abstract and intangible concepts such as birth, happiness, evolution, technology, management, imagination, revenge, politics, hope, cooker, sport, and literacy. Because of the enormous diversity of nouns compared to other parts of speech, the Applicant has found that it is much more relevant to consider the noun phrase as a key linguistic metric. So, the great variety of items classified as nouns by the present invention helps to discriminate and identify individual speech utterances much easier and faster than prior techniques disclosed in the art.

Following this same thought, the present invention also adopts and implements another linguistic entity—the word phrase—to facilitate speech query recognition. The basic structure of a word phrase—whether it be a noun phrase, verb phrase, adjective phrase—is three parts—[pre-Head string], [Head] and [post-Head string]. For example, in the minimal noun phrase—"the children," "children" is classified as the Head of the noun phrase. In summary, because of the diversity and frequency of noun phrases, the choice of noun phrase as the metric by which stored answer is linguistically chosen, has a solid justification in applying this technique to the English natural language as well as other natural languages. So, in sum, the total noun phrases in a speech utterance taken together operate extremely well as unique type of speech query fingerprint.

The ID corresponding to the best answer corresponding to the selected final result record question is then generated by routine **815** which then returns it to DB Process shown in FIG. **4**C. As seen there, a Best Answer ID I is received by routine **716**A, and used by a routine **716**B to retrieve an 65 answer file path. Routine **716**C then opens and reads the answer file, and communicates the substance of the same to

34

routine 716D. The latter then compresses the answer file data, and sends it over data link 160 to client side system 150 for processing as noted earlier (i.e., to be rendered into audible feedback, visual text/graphics, etc.). Again, in the context of a learning/instructional application, the answer file may consist solely of a single text phrase, but in other applications the substance and format will be tailored to a specific question in an appropriate fashion. For instance, an "answer" may consist of a list of multiple entries corresponding to a list of responsive category items (i.e., a list of books to a particular author) etc. Other variations will be apparent depending on the particular environment.

Natural Language Engine 190

Again referring to FIG. 4D, the general structure of NL engine 190 is depicted. This engine implements the word analysis or morphological analysis of words that make up the user's query, as well as phrase analysis of phrases extracted from the query.

As illustrated in FIG. 9, the functions used in a morphological analysis include tokenizers 802A, stemmers 804A and morphological analyzers 806A. The functions that comprise the phrase analysis include tokenizers, taggers and groupers, and their relationship is shown in FIG. 8.

Tokenizer 802A is a software module that functions to break up text of an input sentence 801A into a list of tokens 803A. In performing this function, tokenizer 802A goes through input text 801A and treats it as a series of tokens or useful meaningful units that are typically larger than individual characters, but smaller than phrases and sentences. These tokens 803A can include words, separable parts of word and punctuation. Each token 803A is given an offset and a length. The first phase of tokenization is segmentation, which extracts the individual tokens from the input text and keeps track of the offset where each token originated from in the input text. Next, categories are associated with each token, based on its shape. The process of tokenization is well-known in the art, so it can be performed by any convenient application suitable for the present invention.

Following tokenization, a stemmer process **804**A is executed, which can include two separate forms—inflectional and derivational, for analyzing the tokens to determine their respective stems **805**A. An inflectional stemmer recognizes affixes and returns the word which is the stem. A derivational stemmer on the other hand recognizes derivational affixes and returns the toot word or words. While stemmer **804**A associates an input word with its stem, it does not have parts of speech information. Analyzer **806**B takes a word independent of context, and returns a set of possible parts of speech **806**A.

As illustrated in FIG. 8, phrase analysis 800 is the next step that is performed after tokenization. A tokenizer 802 generates tokens from input text 801. Tokens 803 are assigned to parts of a speech tag by a tagger routine 804, and a grouper routine 806 recognizes groups of words as phrases of a certain syntactic type. These syntactic types include for example the noun phrases mentioned earlier, but could include other types if desired such as verb phrases and adjective phrases. Specifically, tagger 804 is a parts-ofspeech disambiguator, which analyzes words in context. It has a built-in morphological analyzer (not shown) that allows it to identify all possible parts of speech for each token. The output of tagger 804 is a string with each token tagged with a parts-of-speech label 805. The final step in the linguistic process 800 is the grouping of words to form phrases 807. This function is performed by the grouper 806,

and is very dependent, of course, on the performance and output of tagger component 804.

Accordingly, at the end of linguistic processing 800, a list of noun phrases (NP) 807 is generated in accordance with the user's query utterance. This set of NPs generated by NLE 5190 helps significantly to refine the search for the best answer, so that a single-best answer can be later provided for the user's question.

The particular components of NLE **190** are shown in FIG. **4**D, and include several components. Each of these components implement the several different functions required in NLE **190** as now explained.

Initialize Grouper Resources Object and the Library 900—this routine initializes the structure variables required to create grouper resource object and library. Specifically, it initializes a particular natural language used by NLE 190 to create a Noun Phrase, for example the English natural language is initialized for a system that serves the English language market. In turn, it also creates the objects (routines) required for Tokenizer, Tagger and Grouper (discussed above) with routines 900A, 900B, 900C and 900D respectively, and initializes these objects with appropriate values. It also allocates memory to store all the recognized Noun Phrases for the retrieved question pairs.

Tokenizing of the words from the given text (from the ²⁵ query or the paired questions) is performed with routine **909**B—here all the words are tokenized with the help of a local dictionary used by NLE **190** resources. The resultant tokenized words are passed to a Tagger routine **909**C. At routine **909**C, tagging of all the tokens is done and the output ³⁰ is passed to a Grouper routine **909**D.

The Grouping of all tagged token to form NP list is implemented by routine 909D so that the Grouper groups all the tagged token words and outputs the Noun Phrases.

Un-initializing of the grouper resources object and freeing of the resources, is performed by routines 909EA, 909EB and 909EC. These include Token Resources, Tagger Resources and Grouper Resources respectively. After initialization, the resources are freed. The memory that was used to store all Noun Phrases are also de-allocated.

Additional Embodiments

In a e-commerce embodiment of the present invention as illustrated in FIG. 13, a web page 1300 contains typical visible links such as Books 1310, Music 1320 so that on 45 clicking the appropriate link the customer is taken to those pages. The web page may be implemented using HTML, a Java applet, or similar coding techniques which interact with the user's browser. For example, if customer wants to buy an album C by Artist Albert, he traverses several web pages as 50 follows: he first clicks on Music (FIG. 13, 1360), which brings up page 1400 where he/she then clicks on Records (FIG. 14, 1450). Alternatively, he/she could select CDs 1460, Videos 1470, or other categories of books 1410, music 1420 or help 1430. As illustrated in FIG. 15, this brings up 55 another web page 1500 with links for Records 1550, with sub-categories—Artist 1560, Song 1570, Title 1580, Genre 1590. The customer must then click on Artist 1560 to select the artist of choice. This displays another web page 1600 as illustrated in FIG. 16. On this page the various artists 1650 are listed as illustrated—Albert 1650, Brooks 1660, Charlie 1670, Whyte 1690 are listed under the category Artists 1650. The customer must now click on Albert 1660 to view the albums available for Albert. When this is done, another web page is displayed as shown in FIG. 17. Again this web page 65 1700 displays a similar look and feel, but with the albums available 1760, 1770, 1780 listed under the heading Tides

36

1750. The customer can also read additional information 1790 for each album. This album information is similar to the liner notes of a shrink-wrapped album purchased at a retail store. One Album A is identified, the customer must click on the Album A 1760. This typically brings up another text box with the information about its availability, price, shipping and handling charges etc.

When web page 1300 is provided with functionality of a NLQS of the type described above, the web page interacts with the client side and server side speech recognition modules described above. In this case, the user initiates an inquiry by simply clicking on a button designated Contact Me for Help 1480 (this can be a link button on the screen, or a key on the keyboard for example) and is then told by character 1440 about how to elicit the information required. If the user wants Album A by artist Albert, the user could articulate "Is Album A by Brooks available?" in much the same way they would ask the question of a human clerk at a brick and mortar facility. Because of the rapid recognition performance of the present invention, the user's query would be answered in real-time by character 1440 speaking out the answer in the user's native language. If desired, a readable word balloon 1490 could also be displayed to see the character's answer and so that save/print options can also be implemented. Similar appropriate question/answer pairs for each page of the website can be constructed in accordance with the present teachings, so that the customer is provided with an environment that emulates a normal conversational human-like question and answer dialog for all aspects of the web site. Character 1440 can be adjusted and tailored according to the particular commercial application, or by the user's own preferences, etc. to have a particular voice style (man, woman, young, old, etc.) to enhance the customer's experience.

In a similar fashion, an articulated user query might be received as part of a conventional search engine query, to locate information of interest on the INTERNET in a similar manner as done with conventional text queries. If a reasonably close question/answer pair is not available at the server side (for instance, if it does not reach a certain confidence level as an appropriate match to the user's question) the user could be presented with the option of increasing the scope so that the query would then be presented simultaneously to one or more different NLEs across a number of servers, to improve the likelihood of finding an appropriate matching question/answer pair. Furthermore, if desired, more than one "match" could be found, in the same fashion that conventional search engines can return a number of potential "hits" corresponding to the user's query. For some such queries, of course, it is likely that real-time performance will not be possible (because of the disseminated and distributed processing) but the advantage presented by extensive supplemental question/answer database systems may be desirable for some users.

It is apparent as well that the NLQS of the present invention is very natural and saves much time for the user and the e-commerce operator as well. In an e-support embodiment, the customer can retrieve information quickly and efficiently, and without need for a live customer agent. For example, at a consumer computer system vendor related support site, a simple diagnostic page might be presented for the user, along with a visible support character to assist him/her. The user could then select items from a "symptoms" page (i.e., a "monitor" problem, a "keyboard" problem, a "printer" problem, etc.) simply by articulating such symptoms in response to prompting from the support character. Thereafter, the system will direct the user on a

37

real-time basis to more specific sub-menus, potential solutions, etc. for the particular recognized complaint. The use of a programmable character thus allows the web site to be scaled to accommodate a large number of hits or customers without any corresponding need to increase the number of 5 human resources and its attendant training issues.

As an additional embodiment, the searching for information on a particular web site may be accelerated with the use of the NLQS of the present invention. Additionally, a significant benefit is that the information is provided in a 10 user-friendly manner through the natural interface of speech. The majority of web sites presently employ lists of frequently asked questions which the user typically wades item by item in order to obtain an answer to a question or issue. For example, as displayed in FIG. 13, the customer clicks on 15 Help 1330 to initiate the interface with a set of lists. Other options include computer related items at 1370 and frequently asked questions (FAQ) at 1380.

As illustrated in FIG. 18, a web site plan for typical web page is displayed. This illustrates the number of pages that 20 have to be traversed in order to reach the list of Frequently-Asked Questions. Once at this page, the user has to scroll and manually identify the question that matches his/her query. This process is typically a laborious task and may or may not yield the information that answers the user's query. 25 The present art for displaying this information is illustrated in FIG. 18. This figure identifies how the information on a typical web site is organized: the Help link (FIG. 13, 1330) typically shown on the home page of the web page is illustrated shown on FIG. 18 as 1800. Again referring to 30 FIG. 18, each sub-category of information is listed on a separate page. For example, 1810 lists sub-topics such as 'First Time Visitors', 'Search Tips', 'Ordering', 'Shipping', 'Tour Account' etc. Other pages deal with 'Account information' 1860, 'Rates and Policies' 1850 etc. Down another 35 level, there are pages that deal exclusively with a sub-sub topics on a specific page such as 'First Time Visitors' 1960, 'Frequently Asked Questions' 1950, 'Safe Shopping Guarantee' 1940, etc. So if a customer has a query that is best answered by going to the Frequently Asked Questions link, 40 he or she has to traverse three levels of busy and cluttered screen pages to get to the Frequently Asked Questions page 1950. Typically, there are many lists of questions 1980 that have to be manually scrolled through. While scrolling visually, the customer then has to visually and mentally match 45 his or her question with each listed question. If a possible match is sighted, then that question is clicked and the answer then appears in text form which then is read.

In contrast, the process of obtaining an answer to a question using a web page enabled with the present NLQS 50 can be achieved much less laboriously and efficiently. The user would articulate the word "Help" (FIG. 13, 1330). This would immediately cause a character (FIG. 13, 1340) to appear with the friendly response "May I be of assistance. Please state your question?". Once the customer states the 55 question, the character would then perform an animation or reply "Thank you, I will be back with the answer soon". After a short period time (preferably not exceeding 5-7 seconds) the character would then speak out the answer to the user's question. As illustrated in FIG. 18 the answer 60 would be the answer 1990 returned to the user in the form of speech is the answer that is paired with the question 1950. For example, the answer 1990: "We accept Visa, MasterCard and Discover credit cards", would be the response to the query 2000 "What forms of payments do you accept?"

Another embodiment of the invention is illustrated in FIG. 12. This web page illustrates a typical website that

38

employs NLQS in a web-based learning environment. As illustrated in FIG. 12, the web page in browser 1200, is divided into two or more frames. A character 1210 in the likeness of an instructor is available on the screen and appears when the student initiates the query mode either by speaking the word "Help" into a microphone (FIG. 2, 215) or by clicking on the link 'Click to Speak' (FIG. 12, 1280). Character 1210 would then prompt the student to select a course 1220 from the drop down list 1230. If the user selects the course 'CPlusPlus', the character would then confirm verbally that the course "CPlusPlus" was selected. The character would then direct the student to make the next selection from the drop-down list 1250 that contains the selections for the chapters 1240 from which questions are available. Again, after the student makes the selection, the character 1210 confirms the selection by speaking. Next character 1210 prompts the student to select 'Section' 1260 of the chapter from which questions are available from the drop down list 1270. Again, after the student makes the selection, character 1210 confirms the selection by articulating the 'Section' 1260 chosen. As a prompt to the student, a list of possible questions appear in the list box 1291. In addition, tips 1290 for using the system are displayed. Once the selections are all made, the student is prompted by the character to ask the question as follows: "Please ask your query now". The student then speaks his query and after a short period of time, the character responds with the answer preceded by the question as follows: "The answer to your question . . . is as follows: . . . ". This procedure allows the student to quickly retrieve answers to questions about any section of the course and replaces the tedium of consulting books, and references or indices. In short, it is can serve a number of uses from being a virtual teacher answering questions on-the-fly or a flash card substitute.

From preliminary data available to the inventors, it is estimate that the system can easily accommodate 100–250 question/answer pairs while still achieving a real-time feel and appearance to the user (i.e., less than 10 seconds of latency, not counting transmission) using the above described structures and methods. It is expected, of course, that these figures will improve as additional processing speed becomes available, and routine optimizations are employed to the various components noted for each particular environment.

Again, the above are merely illustrative of the many possible applications of the present invention, and it is expected that many more web-based enterprises, as well as other consumer applications (such as intelligent, interactive toys) can utilize the present teachings. Although the present invention has been described in terms of a preferred embodiment, it will be apparent to those skilled in the art that many alterations and modifications may be made to such embodiments without departing from the teachings of the present invention. It will also be apparent to those skilled in the art that many aspects of the present discussion have been simplified to give appropriate weight and focus to the more germane aspects of the present invention. The microcode and software routines executed to effectuate the inventive methods may be embodied in various forms, including in a permanent magnetic media, a non-volatile ROM, a CD-ROM, or any other suitable machine-readable format. Accordingly, it is intended that the all such alterations and modifications be included within the scope and spirit of the invention as defined by the following claims.

What is claimed is:

1. A speech-enabled internet website operating on a server computing system and comprising:

30

- a receiving routine executing on the server computing system for receiving speech data associated with a user speech-based query, said speech data being characterized by a data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and
- a speech recognition routine executing on the server computing system for completing recognition of said speech query using said speech data and said data content to generate a recognized speech query; and
- a web page having a list of items, at least some of said list of items being selectable by a user based on said recognized speech query;
- wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as needed based on computing resources available to said client platform and server computing system respectively.
- 2. The website of claim 1, wherein said web page displays an additional list of one or more items based on said recognized speech query.
- 3. The website of claim 1, wherein said website is adapted so that the user can navigate and locate information of 25 interest using said speech query.
- **4**. The website of claim **1**, wherein said list of items include products and/or services offered by said website.
- **5**. The website of claim **1**, wherein said web page is implemented in HTML or as a Java applet.
- 6. The website of claim 1, wherein said website is further adapted to respond to a speech query concerning said list of items by returning a text or speech articulated response.
- 7. The website of claim 1, wherein said website is further adapted to interact on a real-time basis in response to one or more continuous speech queries.
- 8. The website of claim 1, wherein said speech recognition routine can complete recognition of said speech query with less latency than would that resulting if said additional data content were generated by a client platform used by the user.
- **9**. The website of claim **1**, wherein said data content constitutes a minimum amount of information that can be used by said speech recognition engine to complete accurate 45 recognition of words and sentences in said speech query.
- 10. The website of claim 1, wherein the website also controls an interactive character agent presented to the user for assisting in handling said speech query.
- 11. The system of claim 10, wherein said interactive ⁵⁰ character agent provides suggestions for queries which the user can articulate.
- 12. The system of claim 10, wherein a different interactive character agent can be presented to different users providing speech utterances received by the server computing system.
- 13. The system of claim 10, wherein said interactive character agent is configured to perform a dialog of successive questions and answers with the user during an interactive session.
- 14. The system of claim 10, wherein said server computing system causes said interactive character agent to respond in real-time whenever the user provides selected speech input.
- **15**. The website of claim **1**, wherein said list of items 65 correspond to topics associated with an interactive lesson tutorial.

40

- 16. The system of claim 1 wherein respective signal processing functions to be performed by the client platform and the server computing system are specified by an initialization routine.
- 17. The system of claim 16 wherein respective signal processing functions to be performed by the client platform and the server computing system are further specified in accordance with transmission characteristics associated with a communications channel used for said speech data.
- 18. The system of claim 1 wherein the server computing system is adapted to handle a client platform that can include a plurality of different hand held computing devices covering a range of differing respective computing capabilities.
- 19. The system of claim 1, wherein said speech data is formatted by a client device with at least one predetermined character used to designate end of an utterance.
- **20**. The system of claim **19**, wherein said predetermined character is a NULL character.
- 21. The system of claim 1, wherein the server computing system transfers speech related data for the web page using a hypertext transfer protocol (HTTP).
- 22. The system of claim 1, wherein a signal processing function performed by the client platform includes generating at least partial speech observation vectors using mel frequency cepstral coefficients.
- 23. The system of claim 1, wherein a signal processing function performed by the client platform includes at least calibrating speech and silence components of a speech utterance.
- 24. The system of claim 1 wherein the server computing system is further configured to perform a natural language processing operation on said recognized speech query to recognize a meaning of a sentence of words contained therein.
- 25. The system of claim 24 wherein said server computing system includes a plurality of separate natural language engines.
- 26. The system of claim 24 wherein said natural language processing operation is configured to compare a limited set of phrases from said recognized speech query with a separate set of phrases corresponding to predefined valid queries from users.
- 27. The system of claim 24 wherein text from said recognized speech query is presented to both a natural language engine for performing said natural language processing operation as well as to a database for identifying a meaning of said recognized speech query, such that a response can be provided by said database for at least some recognized speech queries before said natural language processing operation is completed.
- **28**. The system of claim **24**, further including a database query engine which performs part of said natural language operation by combining said speech query with search predicates to retrieve from a database a set of one or more potential responsive answers to said speech query.
- 29. The system of claim 1 wherein the server computing system is further configured to dynamically change a speech recognition grammar based on input provided by a user to selections available within said web page.
- 30. The system of claim 29 wherein multiple speech grammars are available and selectable within the web page, and such that speech input provided by the user for an item within the web page using a first grammar dynamically controls which one of a plurality of second grammars is loaded for speech recognition of subsequent speech input by the user.

Case 3:08-cv-00863-MHP

- 31. The system of claim 29 wherein multiple speech grammars are selectable in a hierarchy within the web page, such that speech input provided by the user for an item within a first level menu of the web page using a first grammar dynamically controls which one of a plurality of 5 second grammars at a second level menu of the web page and/or a second web page is loaded for speech recognition.
- 32. The system of claim 1 wherein the server computing system is further configured to dynamically change a speech recognition grammar based on spoken responses provided 10 by a user during a real-time dialogue session conducted with an interactive electronic agent associated with said web page.
- 33. The system of claim 1, wherein said web page includes first tags which are selectable by one of a pointing 15 device or a keyboard, and separate second tags selectable by speech input.
- 34. The system of claim 1, wherein said web page includes tags which can be selected by a pointing device and/or a keyboard and/or speech input.
- 35. The system of claim 1, wherein said web page and associated speech data is communicated to a client device using a hypertext transfer protocol (HTTP).
- 36. The system of claim 1, wherein said server computing $_{25}$ system includes text to speech capability for outputting a response associated with said web page in audible form.
- 37. The system of claim 1, wherein said speech query is recognized by forming a concatenation of words and/or concatenation as a search query for a database.
- 38. The system of claim 1, wherein the user can speak a help command while interacting with any web page maintained by the server computing system to cause an interactive character agent to appear.
- 39. A speech-enabled internet website operating on a server computing system and comprising:
 - a receiving routine executing on the server computing system for receiving speech data associated with a user speech-based query, said speech data being character- 40 ized by a first data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and
 - a speech recognition routine executing on the server computing system for completing recognition of said speech query using said speech data and said first data content to generate a recognized speech query; and
 - a web page having a search engine for locating user selected information of interest, said search engine using a text query that is derived from said recognized speech auery:
 - wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing 55 system as needed based on computing resources available to said client platform and server computing system respectively.
- 40. The website of claim 39, wherein said speech query is processed by more than one server computing system, so that multiple search engines are used for locating said information of interest.
- 41. The website of claim 39, wherein said web page includes a list of one or more items associated with assisting a user to diagnose a product or service problem, and which 65 one or more items are also selectable by a user speech-based query.

- 42. The website of claim 39, wherein said website provides and controls an agent for assisting a user to interact with said website.
- 43. The website of claim 39, wherein said list of items correspond to topics associated with an interactive lesson tutorial.
- 44. A system for enabling a user web browser program to interact with a website using speech utterances, the system comprising:
- a receiving routine for receiving speech data associated with a speech utterance generated at a client platform, said speech data being characterized by a limited speech data content to reduce processing and transmission latencies; and
- a speech recognition routine executing on a server computing system for completing recognition of said speech utterance using said limited speech data content to generate a recognized speech query in real-time; and
- a web page routine for presenting one or more web pages to the user web browser program, wherein data content for said one or more web pages perceived by the user is controlled by said recognized speech query;
- wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as needed based on computing resources available to said client platform and server computing system respectively.
- 45. The system of claim 44, wherein said recognized phrases derived from said speech query and using said 30 speech query can include one of a number of predefined sentences recognizable by said system, and said speech query is recognized by identifying a candidate set of potential sentences from a number of predefined sentences, and then comparing each entry in the candidate set of potential 35 sentences to said speech query to determine a matching recognized sentence.
 - 46. The system of claim 45, wherein said speech utterance is compared against said candidate set of potential sentences by examining noun phrases.
 - 47. The system of claim 45, wherein said candidate set of potential sentences are determined in part by a context dictionary loaded by said sentence recognition circuit in response to an operating environment presented by said system to a user.
 - 48. The system of claim 44, wherein said speech utterance is processed by a natural language engine.
 - 49. The system of claim 44, wherein environment variables experienced by the user within the web browser program are used for recognizing said speech query, such that said environmental variables vary in accordance with a web page being viewed by the user or a selection within a web page made by the user.
 - 50. The website of claim 44, wherein said list of items correspond to topics associated with an interactive lesson tutorial.
 - 51. The system of claim 44, wherein said limited speech data content does not include complete speech observation vectors which must be derived from said limited speech data content and input to said speech recognition routine before said speech utterance can be recognized.
 - 52. The system of claim 44 wherein said limited speech data content comprises speech data that is transmitted continuously while the user is speaking and until silence is detected.
 - 53. A method of interacting with a web-connected server using a client browser program, the method comprising the steps of:

Document 52-4

43

- (a) receiving speech data associated with a speech utterance articulated by a user of the client platform, said speech data being characterized by a limited speech data content to reduce processing and transmission latencies; and
- (b) completing recognition of said speech utterance using said limited speech data content to generate a recognized speech query at the web-connected server in real-time: and
- (c) presenting one or more web pages to the user client 10 web browser program, such that data content for said one or more web pages transmitted to the client browser program is controlled by said recognized speech query;
- (d) allocating signal processing functions required to 15 generate said recognized speech query between a client platform and the server computing system as needed based on computing resources available to said client and server computing systems respectively.
- 54. The method of claim 53 further including a step: 20 performing a natural language processing operation to compare a limited set of phrases extracted from said recognized speech query with a separate set of phrases extracted from predefined valid queries from users.
- 55. The method of claim 53 further including a step: ²⁵ providing an interactive electronic character who provides suggestions for queries which the user can articulate.
- 56. The system of claim 55, further including a step: configuring said interactive character agent to engage in a dialog of successive questions and answers with the user 30 during an interactive session.
- 57. The method of claim 53, further including a step: presenting an interactive character agent to the user in real-time in response to a spoken help command presented while interacting with any web page maintained by the server computing system.
- 58. The method of claim 53, further including a step: configuring said web page as a single page to a browser to allow a user to ask questions concerning any item identified in said database within said single page.
- 59. The method of claim 53, further including a step: forming a concatenation of words and/or phrases derived from said speech query and using said concatenation as a search query for a database.
- **60**. A method of presenting information from a set of one or more web pages associated with a server interacting through a browser program with a client platform, the method comprising the steps of:
 - (a) partially processing a speech utterance at the client 50 platform to generate limited data content speech data, said limited data content speech data being configured to reduce processing and transmission latencies; and
 - (b) completing processing of said speech utterance using said limited speech data content to generate a recognized speech query at the server; and
 - (c) presenting content for the set of one or more web pages to the browser program, under control of said recognized speech query;
 - (d) allocating signal processing functions required to 60 generate said recognized speech query between a client platform and the server computing system as needed based on computing resources available to said client and server computing systems respectively.
- **61**. The method of claim **60**, wherein said limited speech 65 data content does not include complete speech observation vectors which must be derived from said limited speech data

content and input to said speech recognition routine before said speech utterance can be recognized.

- 62. The method of claim 60, wherein the server computing system transfers speech related data for the web page using a hypertext transfer protocol (HTTP).
- 63. The method of claim 60 further including a step: dynamically changing a speech recognition grammar based on input provided by a user to selections available within said web page.
- 64. A speech-enabled internet server computing system comprising:
 - a receiving routine executing on the server computing system for receiving speech data associated with a user speech-based query, said speech data being characterized by a data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and
 - a speech recognition routine executing on the server computing system for completing recognition of said speech query using said speech data and said data content to generate a recognized speech query;
 - wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as needed based on computing resources available to said client platform and server computing system respectively:
 - a natural language routine executing on the server computing system and configured to process said recognized speech query to generate a natural language result in real-time:
 - a web page having a list of items, at least some of said list of items being selectable by a user based on said natural language result;
 - a database coupled to the server computing system for storing predefined answers which correspond to content for said list of items on said web page.
- 65. The speech-enabled internet server computing system of claim 64, wherein said web page contains links to other web pages which can be selected by speech queries.
- 66. The speech-enabled internet server computing system of claim 64, wherein said web page is a single page configured to allow a user to ask questions concerning any item identified in said database within said single page.
- 67. The speech-enabled internet server computing system of claim 64 wherein any and all of said list of items are selectable in a single screen.
- 68. The speech enabled internet server computing system of claim 67, wherein any and all of said list of items are selectable without scrolling through said web page.
- 69. A speech-enabled internet server computing system comprising:
- a receiving routine executing on the server computing system for receiving speech data associated with a user speech-based query, said speech data being characterized by a data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and
- a speech recognition routine executing on the server computing system for completing recognition of said speech query using said speech data and said data content to generate a recognized speech query;
- wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as

45

- needed based on computing resources available to said client platform and server computing system respec-
- a natural language routine executing on the server computing system and configured to process said recog- 5 nized speech query to generate a natural language result based on an analysis of a selected limited set of phrases presented in said recognized speech query;
 - wherein said selected limited set of phrases are configured so that said natural language engine can 10 generate said natural language result in real-time;
 - a web page having a list of items, at least some of said list of items being selectable by a user based on said natural language result;
 - a database coupled to the server computing system for 15 storing content pertaining to said list of items on said web page.
- 70. The speech-enabled internet server computing system of claim 69 wherein said selected limited set of phrases include combinations of words and phrases contained in a 20 grammar used by said speech recognition routine.
- 71. The speech-enabled internet server computing system of claim 69 wherein said selected limited set of phrases are generated dynamically from the recognized speech query.
- 72. The speech-enabled internet server computing system 25 of claim 69 wherein said natural language engine compares said selected limited set of phrases to a set of phrases contained in predefined answers.
- 73. The speech-enabled internet server computing system of claim 69 wherein said natural language engine result is a 30 single best answer.
- 74. A speech-enabled internet server computing system comprising:
 - a receiving routine executing on the server computing system for receiving speech data associated with a user 35 speech-based query, said speech data being characterized by a data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and
 - a speech recognition routine executing on the server 40 computing system for completing recognition of said speech query using said speech data and said data content to generate a recognized speech query;
 - wherein signal processing functions required to generate said recognized speech query can be allocated

46

- between a client platform and the server computing system as needed based on computing resources available to said client platform and server computing system respectively;
- a natural language routine executing on the server computing system and configured to process said recognized speech query to generate a natural language result based on an analysis of a selected limited set of phrases presented in said recognized speech query;
- wherein said selected limited set of phrases are configured so that said natural language engine can generate said natural language result and a response can be provided to said user speech-based query in
- a web page having a list of items, at least some of said list of items being selectable by a user based on said natural language result;
- a database coupled to the server computing system for storing content pertaining to said list of items on said web page;
- an electronic conversational agent adapted to interact with the user and mimic behavior of a human agent through a native language interactive real-time dialog session with the user.
- 75. The speech-enabled internet server computing system of claim 74, wherein said electronic conversational agent is presented within a client browser.
- 76. The speech-enabled internet server computing system of claim 74, wherein said electronic conversational agent is a visual character on a screen.
- 77. The speech-enabled internet server computing system of claim 74, wherein said electronic conversational agent is configured to articulate suggestions to the user for appropriate speech queries.
- 78. The speech-enabled internet server computing system of claim 74, wherein said electronic conversational agent is adapted to have configurable perception parameters which are adjusted and tailored to said content pertaining to said list of items.

KENT DECLARATION EXHIBIT 4

US007277854B2

(12) United States Patent

Bennett et al.

(10) Patent No.: US 7,277,854 B2

(45) **Date of Patent:** Oct. 2, 2007

(54) SPEECH RECOGNITION SYSTEM INTERACTIVE AGENT

(75) Inventors: Ian M. Bennett, Palo Alto, CA (US);

Bandi Ramesh Babu, Anantapur (IN); Kishor Morkhandikar, Gulbarga (IN); Pallaki Gururaj, Bangalore (IN)

(73) Assignee: Phoenix Solutions, Inc, Palo Alto, CA

(US)

(*) Notice: Subject to any disclaimer, the term of this

patent is extended or adjusted under 35 U.S.C. 154(b) by 82 days.

0.5.C. 154(b) by 62 day

(21) Appl. No.: 11/031,633(22) Filed: Jan. 7, 2005

(65) Prior Publication Data

US 2005/0144004 A1 Jun. 30, 2005

Related U.S. Application Data

- (63) Continuation of application No. 10/684,357, filed on Oct. 10, 2003, which is a continuation of application No. 09/439,145, filed on Nov. 12, 1999, now Pat. No. 6.633,846.
- (51) Int. Cl. G10L 15/18 (2006.01) G06F 17/20 (2006.01) G06F 17/28 (2006.01)

(56) References Cited

U.S. PATENT DOCUMENTS

4,473,904 A 9/1984 Suehiro et al.

4,587,670	\mathbf{A}	5/1986	Levinson et al.	
4,783,803	A	11/1988	Baker et al.	
4,785,408	A	11/1988	Britton et al.	
4,852,170	\mathbf{A}	7/1989	Bordeaux	
4,868,750	Α	* 9/1989	Kucera et al	704/8
4,914,590	\mathbf{A}	4/1990	Loatman et al.	
4,991,094	Α	2/1991	Fagan et al.	
4,991,217	Α	2/1991	Garrett et al.	
5,068,789	\mathbf{A}	11/1991	van Vliembergen	
5,146,405	A	9/1992	Church	
5,157,727	\mathbf{A}	10/1992	Schloss	

(Continued)

FOREIGN PATENT DOCUMENTS

EP 1094388 4/2001

(Continued)

OTHER PUBLICATIONS

B. Arons, "The Design of Audio Servers and Toolkits for Supporting Speech in the User Interface," believed to be published in: Journal of the American Voice I/O Society, pp. 27-41, Mar. 1991.

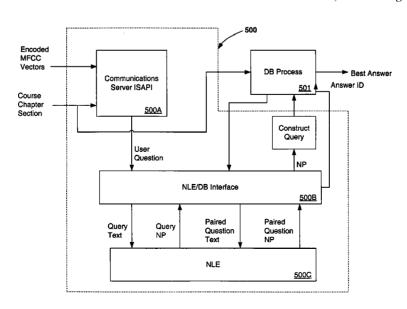
(Continued)

Primary Examiner—Martin Lerner (74) Attorney, Agent, or Firm—J. Nicholas Gross

(57) ABSTRACT

A speech recognition system includes distributed processing across a client and server for recognizing a spoken query by a user. A number of different speech models for different languages are used to support and detect a language spoken by a user. In some implementations an interactive electronic agent responds in the user's language to facilitate a real-time, human like dialogue.

29 Claims, 31 Drawing Sheets



US 7,277,854 B2 Page 2

U.S. PATENT	DOCUMENTS	6,122,613 A	9/2000	
- 221 (F2) - F/1002	C 18 1	6,125,284 A		Moore et al.
	Goldhor et al. Haddock et al.	6,125,341 A		Raud et al. Guberman
- , ,	Pedersen et al.	6,138,089 A 6,141,640 A	10/2000	
-,	Brown et al.	6,144,848 A		Walsh et al.
	Reed et al.	6,144,938 A		Surace et al.
5,384,892 A 1/1995		6,173,261 B1		Arai et al.
	Burns et al.	6,173,279 B1		Levin et al.
	Stanford et al.	6,175,634 B1		Graumann
	Kupiec 704/270.1 Lee et al.	6,178,404 B1		Hambleton et al. Balakrishnan et al.
	Stanford et al.	6,182,038 B1 6,182,068 B1	1/2001	
- , ,	Cohen et al.	6,185,535 B1		Hedin et al.
	Waters	6,192,110 B1		Abella et al.
5,553,119 A 9/1996	McAllister et al.	6,195,636 B1	2/2001	Crupi et al.
	Bissonnette et al.	6,216,013 B1		Moore et al.
	Stanford et al.	6,226,610 B1		Keiller et al.
5,625,814 A 4/1997	Linebarger et al.	6,233,559 B1		Balakrishnan Mohri et al.
	Minakami et al.	6,243,679 B1 6,246,986 B1		Ammicht et al.
-,,	Gorin et al.	6,246,989 B1		Polcyn
-, ,	Baker et al.	6,256,607 B1		Digalakis et al.
5,680,628 A 10/1997	Carus et al.	6,269,153 B1	7/2001	Carpenter et al.
	Driscoll	6,269,336 B1		Ladd et al.
	Cook et al.	6,278,973 B1		Chung et al.
, , ,	Harless	6,292,767 B1		Jackson et al.
	Flanagan et al. Bordeaux	6,292,781 B1		Urs et al. Van Tichelen et al.
- / /	Rongley	6,311,159 B1 6,314,402 B1		Monaco et al.
	Nguyen	6,327,343 B1		Epstein et al.
	Salazar et al.	6,327,561 B1		Smith et al.
5,794,193 A 8/1998	Gorin	6,327,568 B1	12/2001	Joost
- , ,	Cohen et al.	6,336,090 B1		Chou et al.
	Fawcett et al.	6,363,349 B1		Urs et al.
-,,	Sarukkai et al. Ho et al.	6,374,219 B1	4/2002	-
	Gorin et al.	6,374,226 B1 6,381,594 B1		Hunt et al. Eichstaedt et al.
	Beattie et al.	6,389,389 B1		Meunier et al.
	Catallo et al.	6,393,403 B1		Majaniemi
5,873,062 A 2/1999	Hansen et al.	6,408,272 B1	6/2002	White et al.
	Noyes 706/55	6,411,926 B1		Chang
5,884,302 A 3/1999		6,418,199 B1		Perrone
	Wong Gould et al.	6,427,063 B1		Cook et al. Weber
	Braden-Harder et al 707/5	6,434,524 B1 6,434,529 B1		Walker et al.
	Ho et al.	6,446,064 B1		Livowsky
	Setlur et al.	6,453,020 B1		Hughes et al.
	Jacobs et al.	6,453,290 B1	9/2002	Jochumson
	Gould et al.	6,499,011 B1		Souvignier et al.
	Barclay et al.	6,499,013 B1	12/2002	
	Liddy et al	6,510,411 B1		Norton et al. Ruber et al.
	Kellner et al.	6,513,037 B1 6,519,562 B1		Phillips et al.
	Kendall et al.	6,522,725 B2	2/2003	
	Nguyen et al.	6,532,444 B1		Weber
	Liddy et al 707/5	6,567,778 B1		Chao Chang et al.
	Ramaswamy et al.	6,574,597 B1		Mohri et al.
	Gilai et al.	6,584,464 B1		Warthen
	Gorin et al. Bates et al.	6,594,269 B1		Polcyn Bjurstrom et al.
- 7	Gillick et al.	6,594,348 B1 6,601,026 B2		Appelt et al.
	Mohri	6,614,885 B2		Polcyn
	Brode et al.	6,615,172 B1	9/2003	Bennett et al.
6,044,266 A 3/2000	Kato	6,633,846 B1		Bennett et al.
	Gorin et al.	6,647,363 B2*	11/2003	
	Martino et al.	6,651,043 B2		Ammicht
	Redfern	6,665,640 B1	12/2003	Bennett et al. Kanevsky et al.
, , ,	de Hita et al. Driscoll	6,665,644 B1 6,681,206 B1		Gorin et al.
	Callan	6,697,780 B1		Beutnagel et al.
- / /	Goldenthal et al.	6,738,743 B2		Sharma et al.
-, ,	Kuhn et al.	6,742,021 B1		Halverson et al.
-, ,				

Page 3

6,785,647	B2	8/2004	Hutchinson	
6,785,654	B2	8/2004	Cyr et al.	
6,823,308	B2	11/2004	Keiller et al.	
6,842,767	B1	1/2005	Partovi et al.	
6,856,960	B1	2/2005	Dragosh et al.	
6,862,713	B1	3/2005	Kraft et al.	
6,871,179	B1	3/2005	Kist et al.	
6,901,399	B1 *	5/2005	Corston et al 70	7/
6,922,733	B1	7/2005	Kuiken et al.	
6,940,953	B1	9/2005	Eberle et al.	
6,941,273	В1	9/2005	Loghmani et al.	
6,944,586	B1	9/2005	Harless et al.	
6,961,954	В1	11/2005	Maybury et al.	
6,964,012	B1	11/2005	Zirngibl et al.	
6,965,864	B1	11/2005	Thrift et al.	
6,965,890	B1	11/2005	Dey et al.	
7,003,463	B1	2/2006	Maes et al.	
7,020,609	B2	3/2006	Thrift et al.	
7,031,925	B1	4/2006	Goldberg	
7,058,573	B1	6/2006	Murveit et al.	
7,082,397	B2	7/2006	Cohen et al.	
2001/0013001	A1	8/2001	Brown et al.	
2001/0016813	A1	8/2001	Brown et al.	
2001/0032083	A1	10/2001	Van Cleven	
2001/0056346	A1	12/2001	Ueyama et al.	
2002/0032566	A1	3/2002	Tzirkel-Hancock et al.	
2002/0046023	A1	4/2002	Fujii et al.	
2002/0059068	A1	5/2002	Rose et al.	
2002/0059069	A1	5/2002	Hsu et al.	
2002/0086269	A1	7/2002	Shpiro	
2002/0087325	A1	7/2002	Lee et al.	
2002/0087655	A1	7/2002	Bridgman et al.	
2002/0091527	A1	7/2002	Shiau	
2003/0191625		10/2003		
2005/0091056		4/2005		
2005/0131704	A1	6/2005	Dragosh et al.	

FOREIGN PATENT DOCUMENTS

EP	1096471	5/2001
WO	WO98/11534	3/1998
WO	WO99/48011	9/1999
WO	WO99/50830	10/1999
WO	WO 00/14727	3/2000
WO	WO 00/17854	3/2000
WO	WO 00/20962	4/2000
WO	WO 00/21075	4/2000
WO	WO 00/21232	4/2000
WO	WO 00/22610	4/2000
WO	WO 00/30072	5/2000
WO	WO 00/30287	5/2000
WO	WO 00/68823	11/2000
WO	WO 01/16936	3/2001
WO	WO 01/18693	3/2001
WO	WO 01/26093	4/2001
WO	WO 01/78065	10/2001
WO	WO 01/95312	12/2001
WO	WO 02/03380	1/2002

OTHER PUBLICATIONS

- L.R. Bahl et al., "A Maximum Likelihood Approach to Continuous Speech Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-5, pp. 179-190, Mar. 1983. (12 pages). J.H. Baker, "The Dragon System—An Overview," IEEE Trans. on ASSP Processing, ASSP-23(1): Feb. 24-29, 1975.
- L.E. Baum et al., "Statistical Inference for Probabilistic Functions for Finite State Markov Chains," The Annals of Mathematical Statistics, 37: 1554-1563, 1966.
- L.E. Baum et al., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," The Annals of Mathematical Statistics, 1970, vol. 41, No. 1, pp. 164-171.

- L.E. Baum, "An Inequality and Associated Maximation Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," Inequalities-III, pp. 1-8, 1972.
- I. Bennett, "A Study of Speech Compression Using Analog Time Domain Sampling techniques," A Dissertation Submitted to the Dept. Of Electrical Engineering and the Committee on Graduate Studies of Stanford University, May 1975, pp. 16-32; 76-111.
- V. Digilakis et al., "Product-Code Vector Quantization of Cepstral Parameters for Speech Recognition over the WWW," believed to be published in: Proc. ICSLP '98, 4 pages, 1998.
- V. Digilakis et al., "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web," believed to be published in: IEEE Journal on Selected Areas of Communications, 22 pages, 1999.
- J.L. Flanagan, "Speech Analysis Synthesis and Perception," 2nd edition, Springer-Verlag Berlin, 1972, pp. 1-53.
- G.D. Forney, "The Viterbi Algorithm," Proceedings of the IEEE, vol. 73, pp. 268-278, Mar. 1973.
- A. Gersho et al., "Vector Quantization and Compression," Kluwer Academic Publishers, 1991, pp. 309-340.
- D. Giuliani et al., "Training of HMM with Filtered Speech Material for Hands-Free Recognition," believed to be published in: Proceedings of ICASSP '99, Phoenix, USA, 4 pages, 1999.
- T. Hazen et al., "Recent Improvements in an Approach to Segment-Based Automatic Language Identification," believed to be published in: Proceedings of the 1994 International Conference on Spoken Language Processing, Yokohama, Japan, pp. 1883-1886, Sep. 1994. D. House, "Spoken-Language Access to Multimedia (SLAM): A Multimodal Interface to the World-Wide Web," Masters Thesis, Oregon Graduate Institute, Department of Computer Science & Engineering, 59 pages, Apr. 1995.
- R.A. Hudson, "Word Grammar," Blackman Inc., Cambridge, MA, 1984, pp. 1-14; 41-42; 76-90; 94-98; 106-109; 211-220.
- F. Jelinek et al, "Continuous Speech Recognition: Statistical Methods," Handbook of Statistics, vol. 2, P.R. Krishnaiah, Ed. Amsterdam, The Netherlands, North-Holland, 1982, pp. 549-573. L. Julia et al., "http://www.speech.sri.com/demos/ atis.html," believed to be published in: Proceedings AAAI'97: Stanford, pp. 72-76, Jul. 1997.
- R. Lau et al, "Webgalaxy-Integrating Spoken Language and Hypertext Navigation," believed to be published in: in Kokkinakis, G. et al., (Eds.) Eurospeech '97, Proceedings of the 5th European Conference on Speech Communication and Technology, Rhodes (Greece), Sep. 22-25, 1997. pp. 883-886, 1997.
- P. Lieberman, "Intonation, Perception and Language," Research Monograph No. 38, MIT Press, Cambridge, Mass., 1967, pp. 5-37. B. Lin et al., "A Distributed Architecture for Cooperative Spoken Dialogue Agents with Coherent Dialogue State and History," believed to be published in: IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado, USA, 4 pages, Dec. 1999.
- B. Lu et al., "Scalability Issues in the Real Time Protocol (RTP)," Project Report for CPSC 663 (Real Time Systems), Dept. of Computer Science, Texas A & M University, 19 pages, 1999.
- J. Makhoul et al., "Vector Quantization in Speech Coding," Proceedings of the IEEE, vol. 73, No. 11, Nov. 1985, pp. 1551-1588. (38 pages).
- H. Melin, "On Word Boundary Detection in Digit-Based Speaker Verification," believed to be published in: Workshop on Speaker Recognition and its Commercial and Forensic Applications (RLA2C), Avignon, France, Apr. 20-23, pp. 46-49, 1998.
- N. Morgan et al., "Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition," Journal of Pattern Recognition and Artificial Intelligence, vol. 7, No. 4, 1993, pp. 899-916.
- R. Quirk et al., "A Comprehensive Grammar of English Language," Longman, London and New York, 1985, pp. 245-331.
- L. Rabiner, "Digital Processing of Speech Signals," Prentice Hall, 1978, pp. 116-171; 355-395.
- L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of the IEEE, vol. 77, No. 2, Feb. 1989, pp. 257-286. (30 pages).

Page 4

L. Rabiner et al., "Fundamentals of Speech Recognition," Prentice Hall, 1993, pp. 11-68.

G. Ramaswamy et al., "Compression of Acoustic Features for Speech Recognition in Network Environments," believed to be published in: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 977-980, Jun. 1998.

L. Travis, "Handbook of Speech Pathology," Appleton-Century-Crofts, Inc., 1957, pp. 91-124.

S. Tsakalidis et al., "Efficient Speech Recognition Using Subvector Quantization and Discrete-Mixture HMMs," believed to be published in: Proc. ICASSP '99, 4 pages, 1999.

Agarwal, R., Towards a PURE Spoken Dialogue System for Information Access, believed to be published in *Proceedings of the ACL/EACL Workshop on Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications*, Madrid, Spain, 1997, 9 pages.

Ammicht, Egbert et al., "Knowledge Collection for Natural Language Spoken Dialog Systems," believed to be published in *Proc. Eurospeech*, vol. 3, p. 1375-1378, Budapest, Hungary, Sep. 1999, 4 pages.

AT&T Corp., "Network Watson 1.0 System Overview," 1998, 4 pages.

AT&T Corp., "AT&T Watson Advanced Speech Applications Platform," 1996, 3 pages.

AT&T Corp., "AT&T Watson Advanced Speech Applications Platform Version 2.0," 1996, 3 pages.

AT&T Corp., "AT&T Watson Advanced Speech Application Platform Version 2.0," 1996, 8 pages.

Gorin, Allen, "Processing of Semantic Information in Fluently Spoken Language," believed to be published in *Proc. ICSLP*, Philadelphia, PA, Oct. 1996, 4 pages.

Gorin, Allen et al., "How May I Help You," believed to be published in *Proc. IVTTA*, Basking Ridge, NJ, Oct. 1996, 32 pages.

Mohri, Mehryar, "String Matching With Automata," Nordic Journal of Computing, 1997, 15 pages.

Prudential News, "Prudential Pilots Revolutionary New Speech-Based Telephone Customer Service System Developed by AT&T Labs—Company Business and Marketing," Dec. 6, 1999, 3 pages. Riccardi, Giuseppe et al., "A spoken language system for automated call routing," believed to be published in *Proc. ICASSP '97*, 1997, 4 pages.

Sharp, Douglas, et al., "The Watson Speech Recognition Engine," accepted by ICASSP, 1997, 9 pages.

European Patent Office search report for EP Application No. 00977144, dated Mar. 30, 2005, 5 pages.

Burstein, A. et al. "Using Speech Recognition In A Personal Communications System," *Proceedings of the International Conference on Communications*, Chicago, Illinois, Jun. 14-18, 1992, pp. 1717-1721.

Digalakis, V. et al., "Quantization Of Cepstral Parameters For Speech Recognition Over The World Wide Web," *IEEE Journal on Selected Areas of Communications*, 1999, pp. 82-90.

Kuhn, T. et al., "Hybrid In-Car Speech Recognition For Mobile Multimedia Applications," *Vehicular Technology Conference*, Houston, Texas, May 1999, pp. 2009-2013.

Cox, Richard V. et al., "Speech and Language Processing for Next-Millennium Communications Services," Proceedings of the IEEE, vol. 88, No. 8, Aug. 2000, pp. 1314-1337.

Kuhn, Roland, and Renato De Mori, "The Application of Semantic Classification Trees to Natural Language Understanding," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, No. 5, May 1995, pp. 449-460.

Unisys Corp., "Natural Language Speech Assistant (NLSA) Capabilities Overview," NLR 3.0, Aug. 1998, Malvern, PA, 27 pages.

Buo, Finn Dag et al., "Learning to parse spontaneous speech." Believed to be published in ICSLP, 1996, 4 pages.

Golden, J. et al., "Automatic Topic Identification for Two-level Call Routing," Proc. International Conference on Acoustics, Speech and Signal Processing, vol. 1, pp. 509-512, 1999.

Carbonell, J. et al., Dynamic strategy selection in flexible parsing.) In Proceedings of the 19th annual meeting of the ACL (ACL 81), 1981, pp. 143-147.

Seneff, S. et al., "Galaxy-II: a Reference Architecture for Conversational System Development," Believed to be published in Proceedings of ICSLP98 (1998), 4 pages.

Microsoft Internet Developer, "Add Natural Language Search Capabilities to Your Site with English Query," http://www.microsoft.com/mind/0498/equery.asp, 1998, 9 pages.

Carroll, J. et al., "Dialogue Management in Vector-Based Call Routing," Believed to be published In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 1998, pp. 256-262.

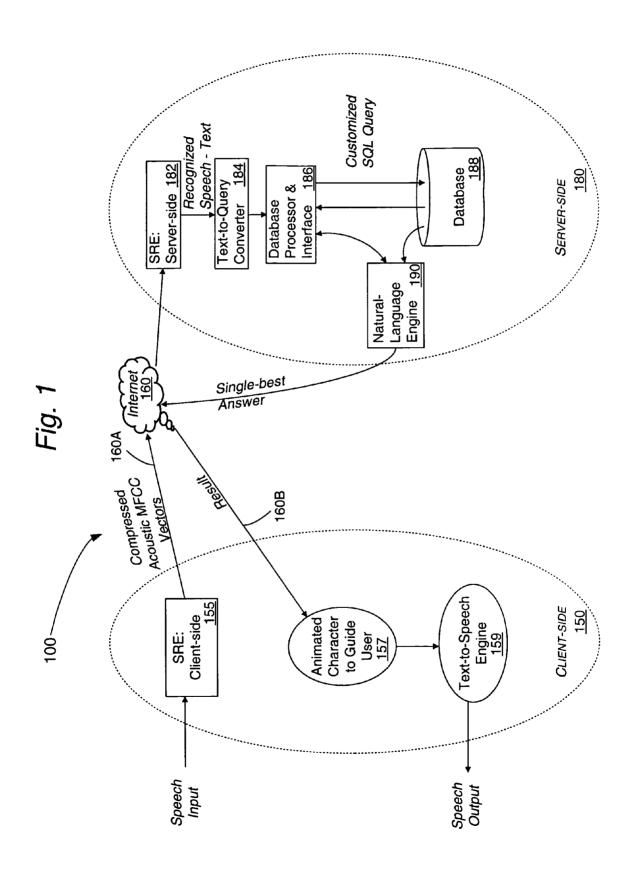
Kaiser, Ed., "Robust, Finite-state Parsing for Spoken Language", Student Session of ACL '99, Jun. 1999, pp. 573-578.

Coffman, Daniel et al., Provisional Application for Patent, U.S. Appl. No. 60/117,595, filed Jan. 27, 1999, 111 pages.

* cited by examiner

Oct. 2, 2007

Sheet 1 of 31



U.S. Patent

Figure 2A

Oct. 2, 2007

Sheet 2 of 31

US 7,277,854 B2

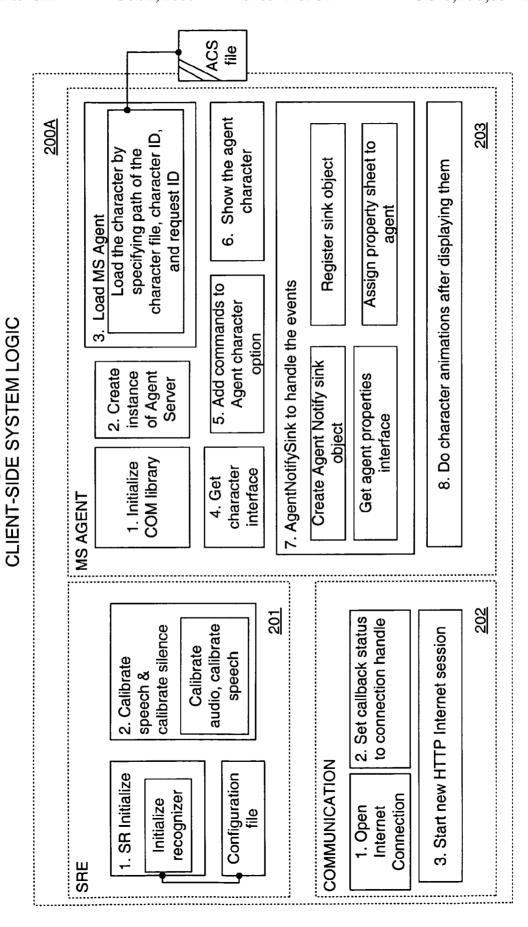


Figure 2B

CLIENT-SIDE SYSTEM LOGIC

Patent

Oct. 2, 2007

Sheet 3 of 31

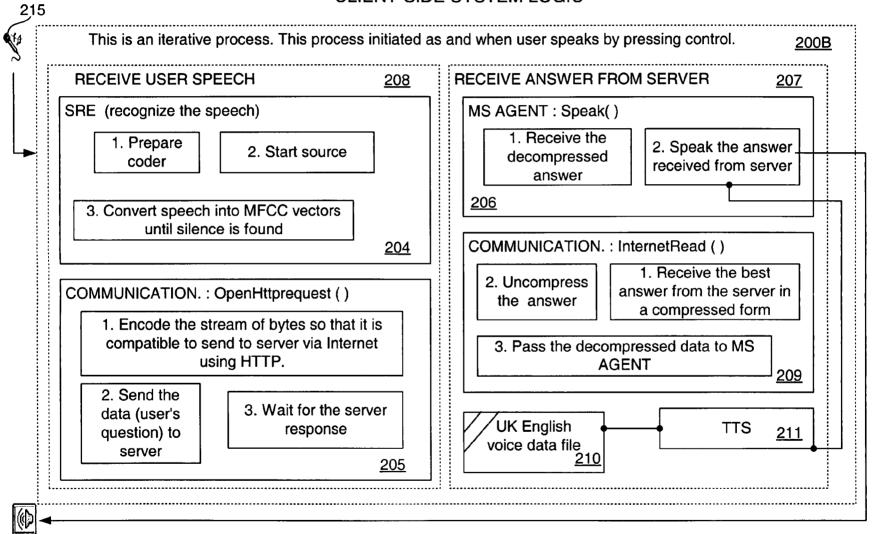
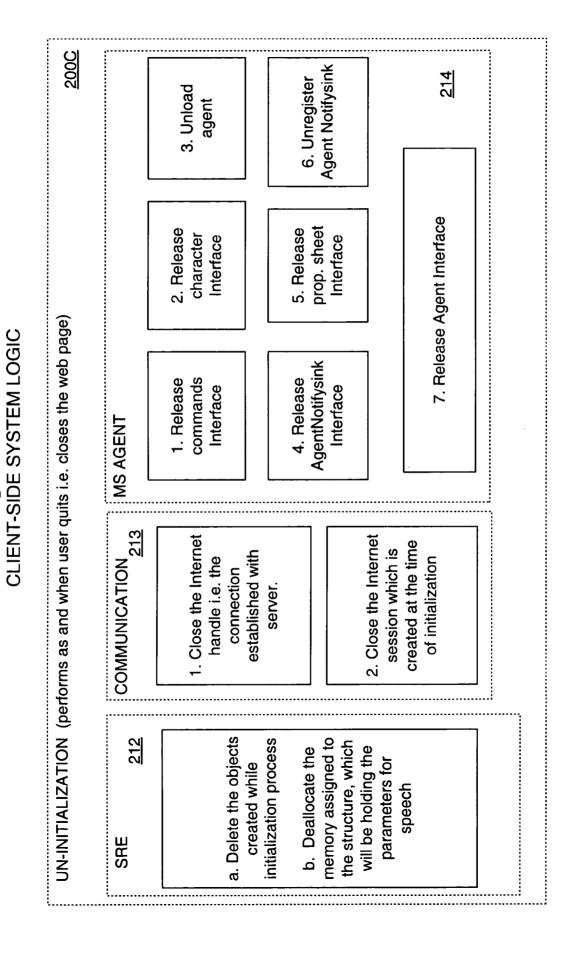


Figure 2C

Oct. 2, 2007

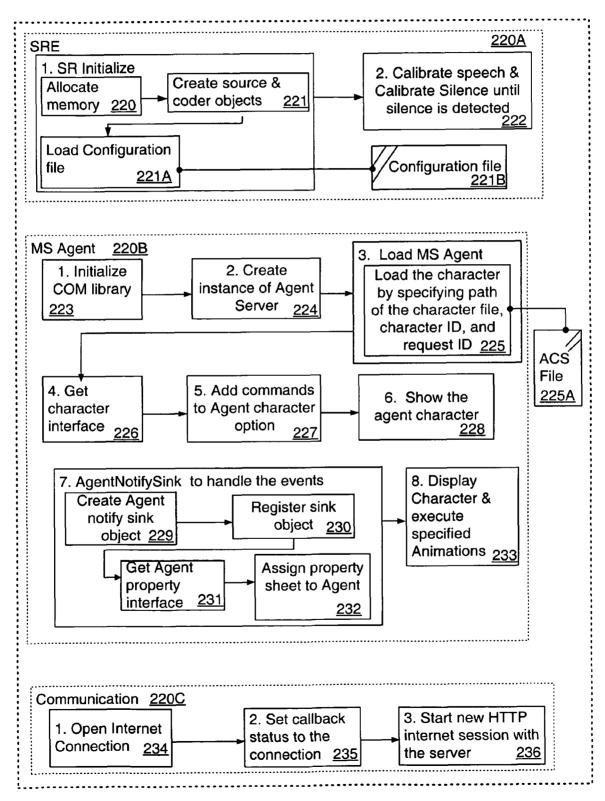
Sheet 4 of 31



Oct. 2, 2007

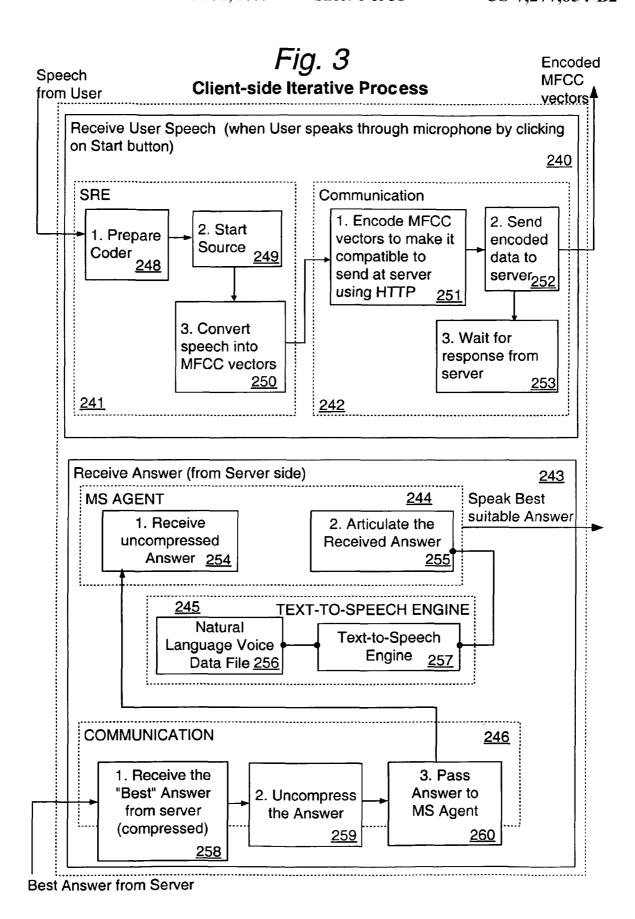
Sheet 5 of 31

Fig. 2D
Client-side Initialization



Oct. 2, 2007

Sheet 6 of 31



Oct. 2, 2007

Sheet 7 of 31

Fig. 4
Client-side Un-Initialization

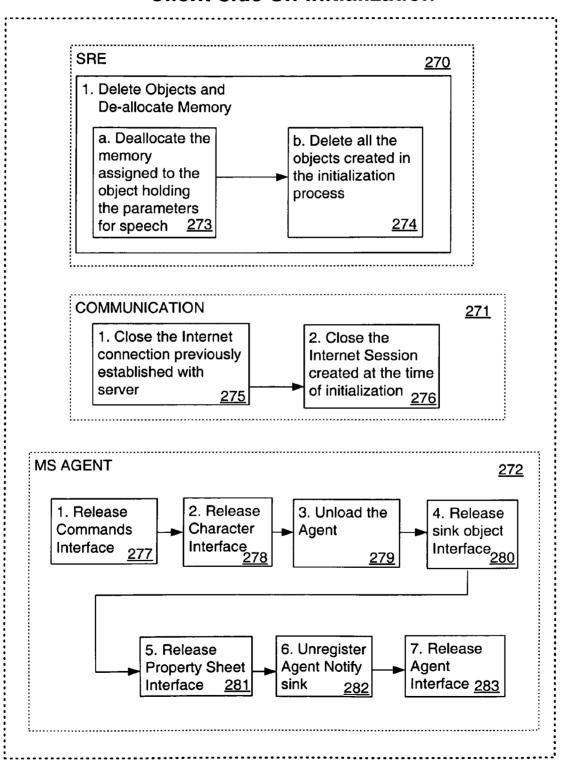
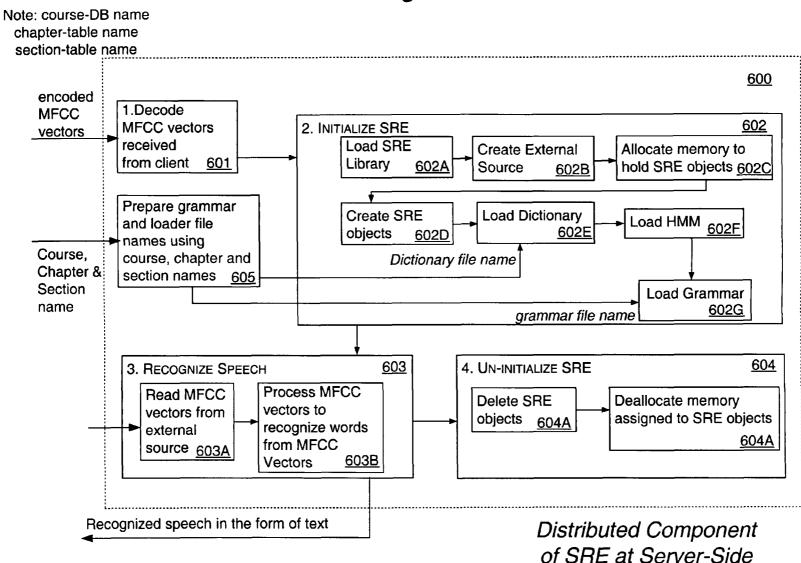
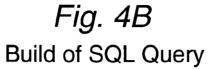


Fig. 4A

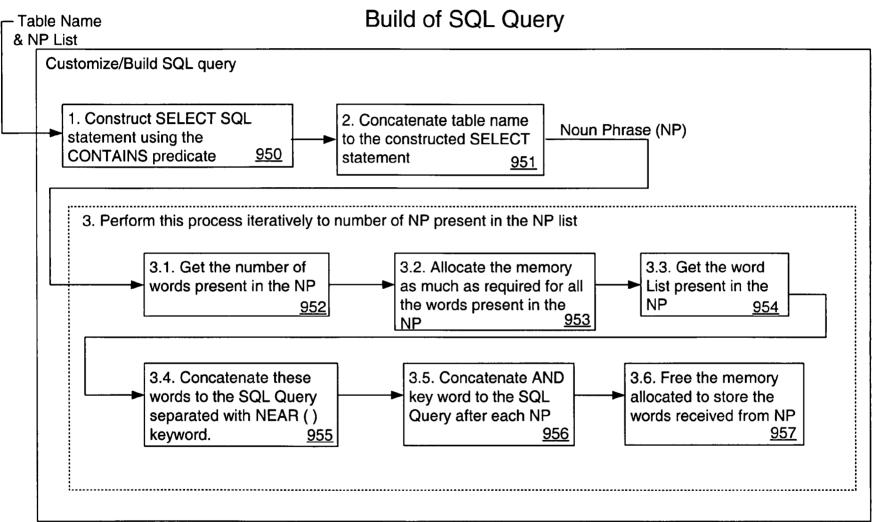




Patent

Oct. 2, 2007

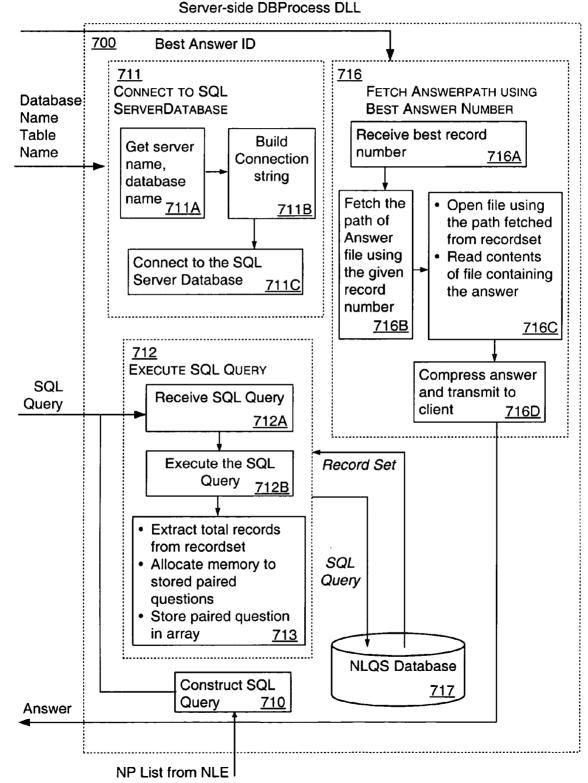
Sheet 9 of 31



Oct. 2, 2007

Sheet 10 of 31

Fig. 4C

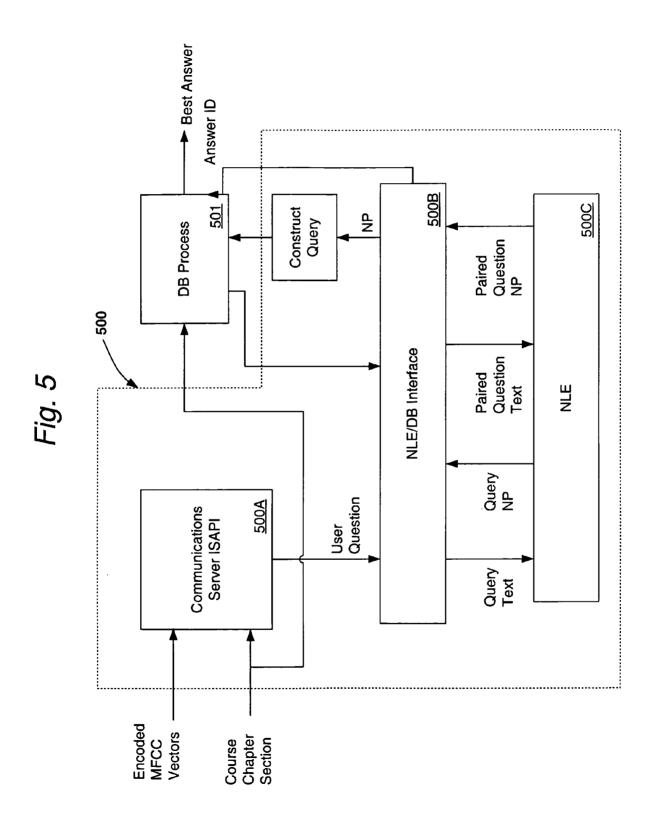


U.S. Patent Oct. 2, 2007 **Sheet 11 of 31** US 7,277,854 B2

Note: PQ - Paired Question Interface Logic between NP- Noun Phrase NLE and DBProcess.DLL Best Red Line - I / O Answer NP list of Number Paired Questions from DB PQ815 813 GET NP LIST **GET BEST ANSWER ID** FOR PAIRED 880 GET NP LIST FOR **≜**Best QUESTION THE USER'S QUESTION Answer NP List from Receive the Number Receive the Question Get the PQs from question ♦and PQ NP list DBProcess.dll from client using 815A Compare NP list 813A **NLE** 880A Compare NP of 880B user's question with Get NP List PQ from DB to find using NLE Question out the best suitable 813B δ question present in NP List of the DB **NP List** User's Question Question **Paired Questions** 9b. Tokenize 9c. Tag all the 900 INITIALIZE GROUPER the words from tokens 909C **RESOURCES** the given text Initialize Initialize 909B Token Tagger Resources Resources 900A 900B 9d. Group all tagged tokens Initialize Create to form the NP Grouper Grouper 909D resources 900C 900D 9E. UN-INITIALIZE GROUPER RESOURCES OBJECT AND FREE THE **RESOURCES** Free token Free grouper Free tagger NLE resources resources resources 909EC 909EB 909EA

Oct. 2, 2007

Sheet 12 of 31



Oct. 2, 2007

Sheet 13 of 31

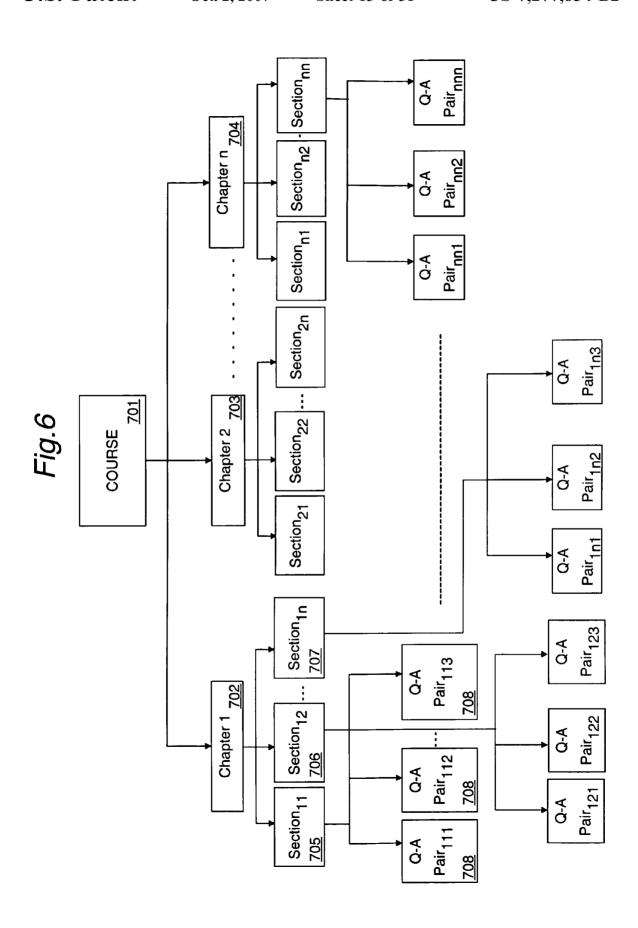


Fig.7A

FIELD NAME <u>701A</u>	Dата Түре <u>702А</u>	Size <u>703A</u>	NULL 704A	PRIMARY KEY 705A	INDEXED? <u>706A</u>
ChapterName <u>707A</u>	Varchar	255	No	No	Yes
SectionName <u>708A</u>	Varchar	255	No	No	Yes

Fig.7B

FIELD NAME 720	Dата Түре <u>721</u>	Size <u>722</u>	NULL <u>723</u>	PRIMARY KEY <u>724</u>	INDEXED? <u>725</u>
Chapter_ID <u>726</u>	Integer		No	Yes	Yes
Answer_ID <u>727</u>	Char	5	No	UNIQUE	Yes
Section_Name <u>728</u>	Varchar	255	No	UNIQUE	Yes
Answer_Title 729	Varchar	255	Yes	No	Yes
PairedQuestion <u>730</u>	Text	16	No	No	Yes (Full-Text)
AnswerPath 731	Varchar	255	No	No	Yes
Creator <u>732</u>	Varchar	50	No	No	Yes
Date_of_Creation 733	Date	-	No	No	Yes
Date_of_Modification 734	Date	<u>-</u>	No	No	Yes

Fig. 7C

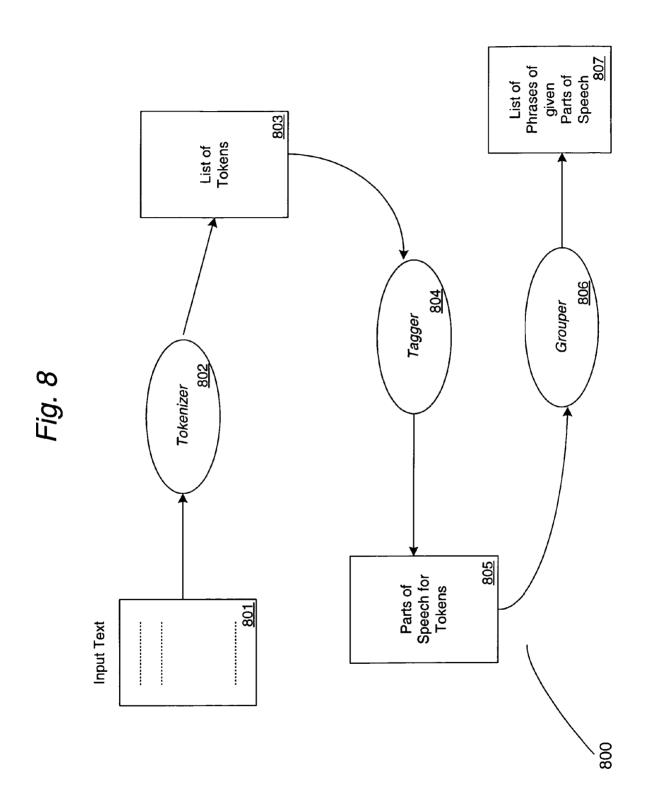
Field	<u>720</u>	Description 735
AnswerID	<u>727</u>	An integer - automatically incremented for user convenience
Section_Name	<u>728</u>	Name of section to which the particular record belongs. This field along with AnswerlD has to be made primary key
Answer_Title	<u>729</u>	A short description of the answer
PairedQuestion	<u>730</u>	Contains one or more combinations of questions for the related answer whose path is stored in the next column AnswerPath
AnswerPath	<u>731</u>	Contains the path of text file, which contains the answer to the related questions stored in the previous column
Creator	<u>732</u>	Name of content creator
Date_of_Creation	<u>733</u>	Date on which content has been added
Date_of_Modification	<u>734</u>	Date on which content has been changed or modified

Fig. 7D

FIELD <u>740</u>	Dата Т үре <u>741</u>	Size <u>742</u>	Null <u>743</u>	PRIMARY KEY <u>744</u>	INDEXED 745
Answer_ID <u>746</u>	Char	5	No	Yes	Yes
Answer_Title 747	Varchar	255	Yes	No	No
PairedQuestion 748	Text	16	No	No	Yes (Full-Text)
Answer_Path 749	Varchar	255	No	No	No
Creator <u>750</u>	Varchar	50	No	No	No
Date_of_Creation 751	Date	_	No	No	No
Date_of_Modification <u>752</u>	Date	-	No	No	No

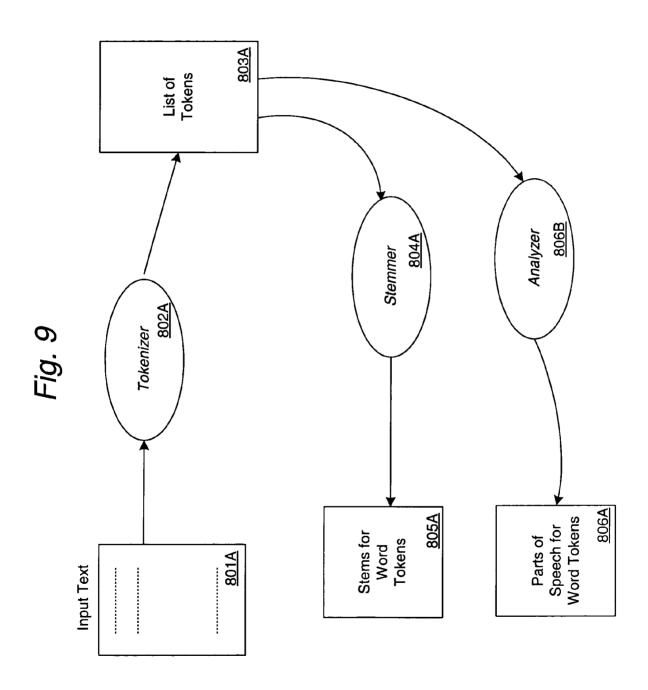
Oct. 2, 2007

Sheet 18 of 31



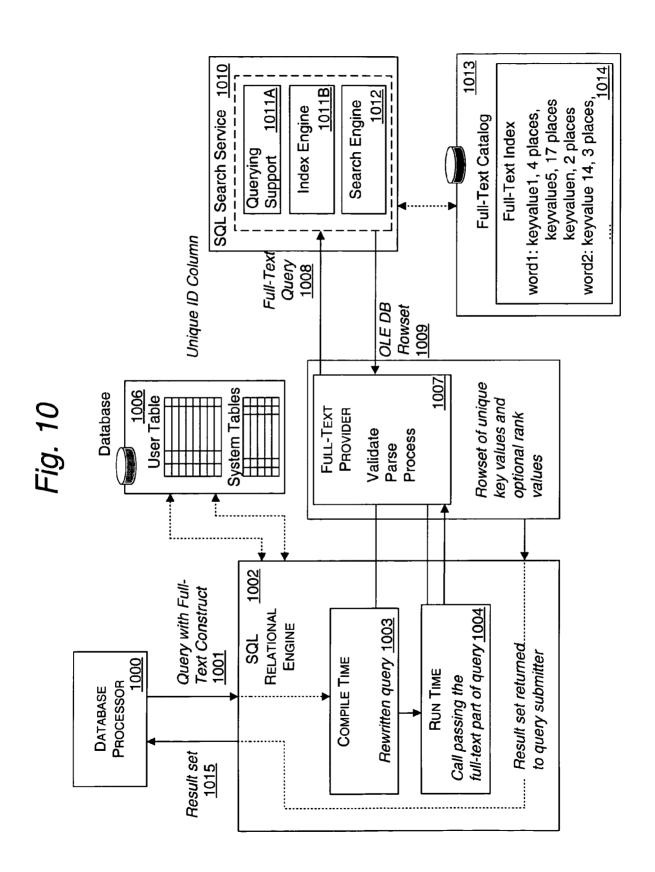
Oct. 2, 2007

Sheet 19 of 31



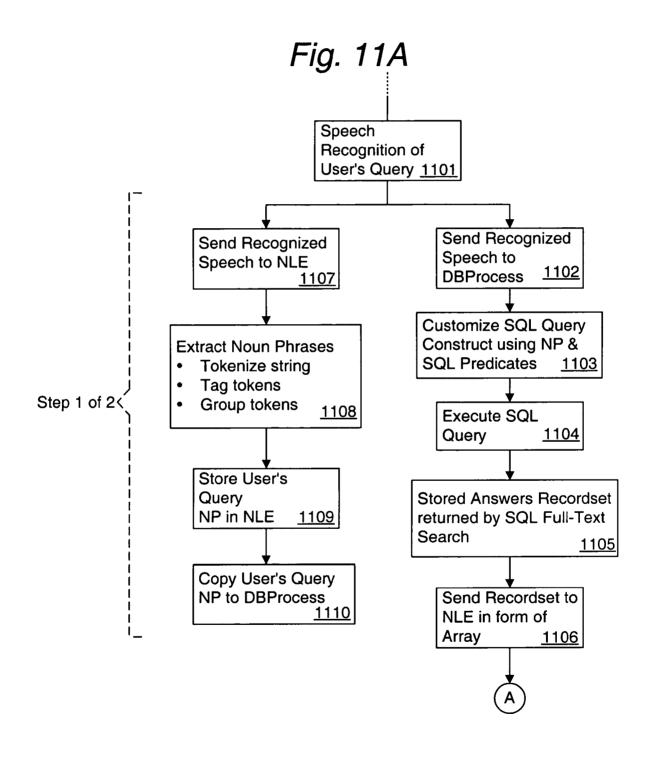
Oct. 2, 2007

Sheet 20 of 31



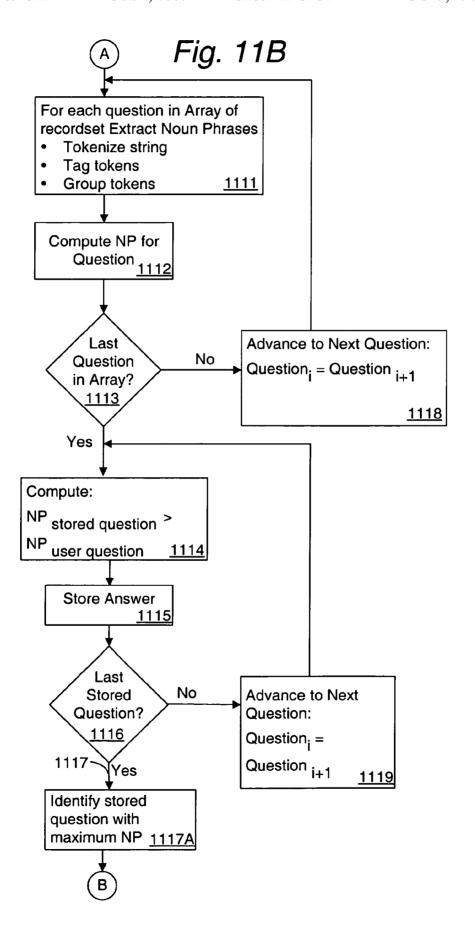
Oct. 2, 2007

Sheet 21 of 31



Oct. 2, 2007

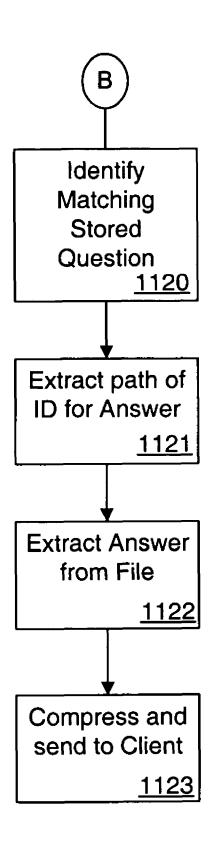
Sheet 22 of 31



Oct. 2, 2007

Sheet 23 of 31

Fig. 11C



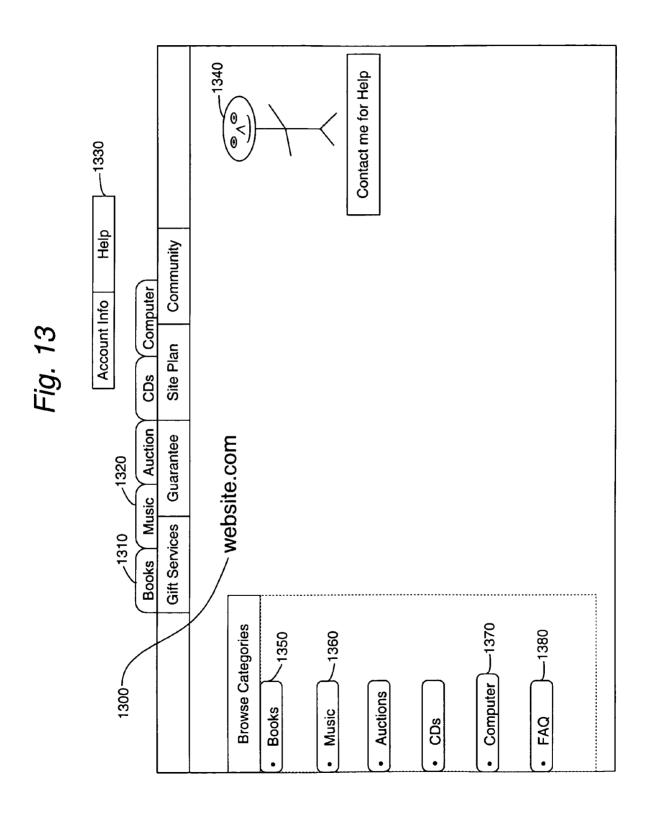
Patent

Oct. 2, 2007

Sheet 24 of 31

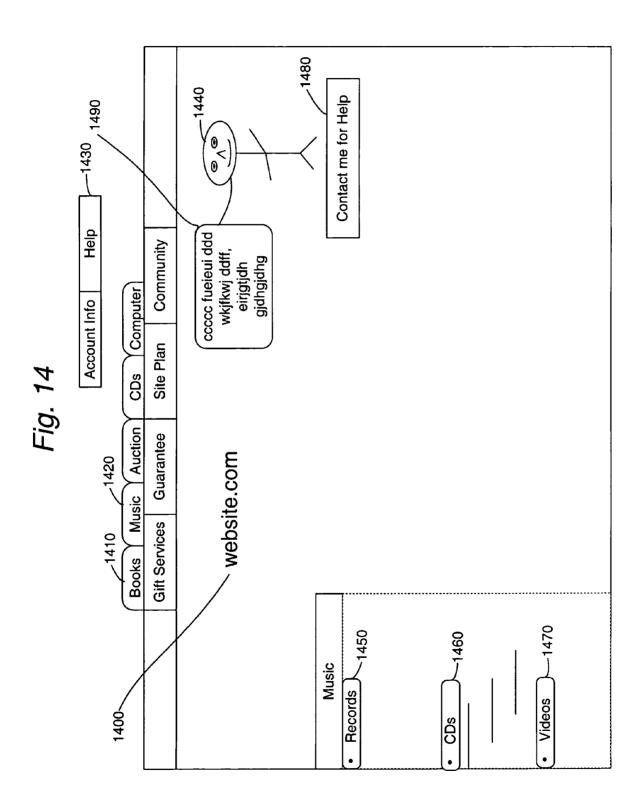
Oct. 2, 2007

Sheet 25 of 31



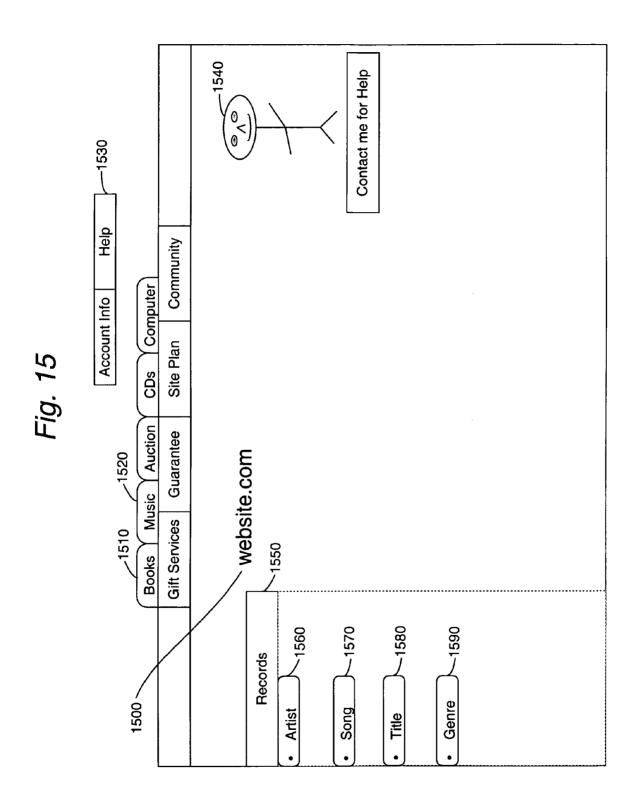
Oct. 2, 2007

Sheet 26 of 31



Oct. 2, 2007

Sheet 27 of 31



Oct. 2, 2007

Sheet 28 of 31

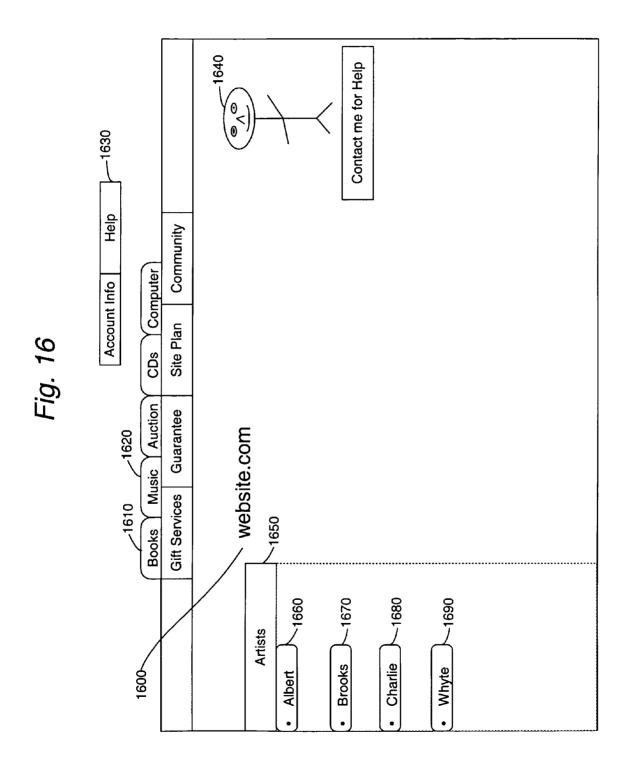
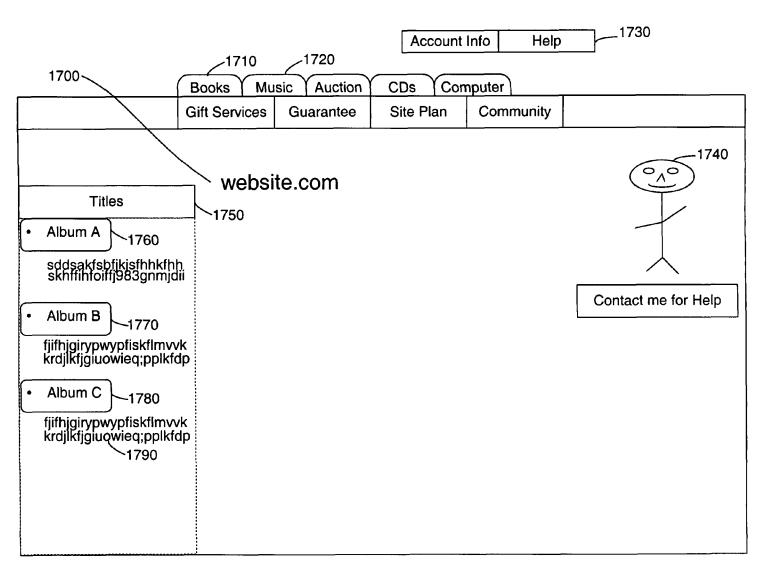
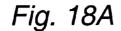


Fig. 17





Patent

Oct. 2, 2007

Sheet 30 of 31

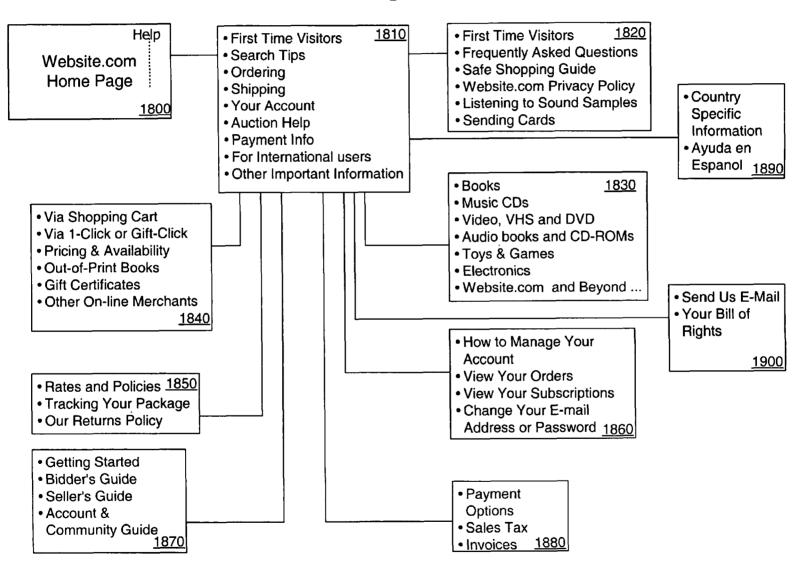
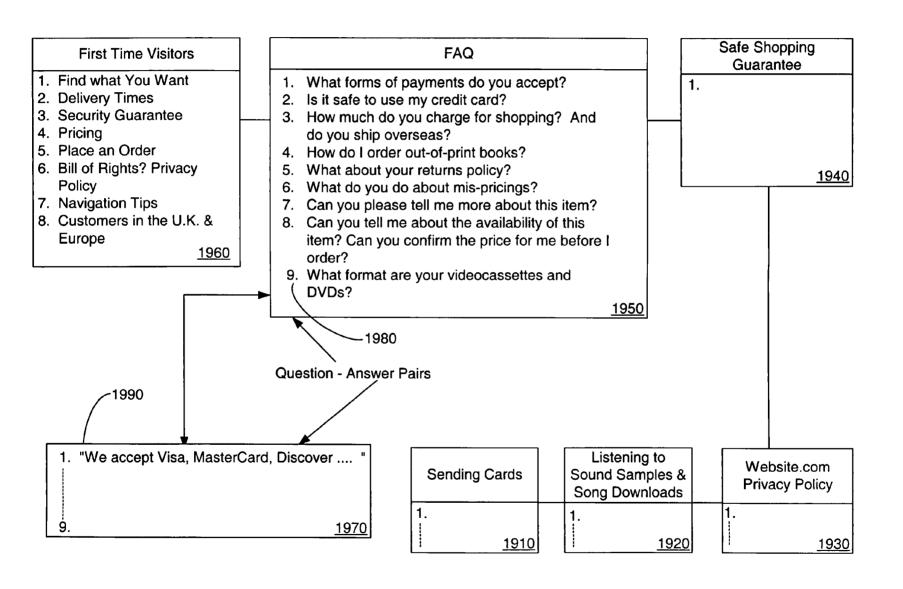


Fig. 18B

Patent

Oct. 2, 2007

Sheet 31 of 31



1

SPEECH RECOGNITION SYSTEM INTERACTIVE AGENT

RELATED APPLICATIONS

The present application claims priority to and is a continuation of Ser. No. 10/684,357 filed Oct. 10, 2003—which in turn is a continuation of Ser. No. 09/439,145 filed Nov. 12, 1999 (now U.S. Pat. No. 6,633,846). Both applications are hereby incorporated by reference herein.

FIELD OF THE INVENTION

The invention relates to a system and an interactive method for responding to speech based user inputs and queries presented over a distributed network such as the INTERNET or local intranet. This interactive system when implemented over the World-Wide Web services (WWW) of the INTERNET, functions so that a client or user can ask a question in a natural language such as English, French, German, Spanish or Japanese and receive the appropriate answer at his or her computer or accessory also in his or her native natural language. The system has particular applicability to such applications as remote learning, e-commerce, technical e-support services, INTERNET searching, etc.

BACKGROUND OF THE INVENTION

The INTERNET, and in particular, the World-Wide Web 30 (WWW), is growing in popularity and usage for both commercial and recreational purposes, and this trend is expected to continue. This phenomenon is being driven, in part, by the increasing and widespread use of personal computer systems and the availability of low cost INTER- 35 NET access.

The emergence of inexpensive INTERNET access devices and high speed access techniques such as ADSL, cable modems, satellite modems, and the like, are expected to further accelerate the mass usage of the WWW.

Accordingly, it is expected that the number of entities offering services, products, etc., over the WWW will increase dramatically over the coming years. Until now, however, the INTERNET "experience" for users has been limited mostly to non-voice based input/output devices, such as keyboards, intelligent electronic pads, mice, trackballs, printers, monitors, etc. This presents somewhat of a bottleneck for interacting over the WWW for a variety of reasons.

First, there is the issue of familiarity. Many kinds of 50 applications lend themselves much more naturally and fluently to a voice-based environment. For instance, most people shopping for audio recordings are very comfortable with asking a live sales clerk in a record store for information on titles by a particular author, where they can be found 55 in the store, etc. While it is often possible to browse and search on one's own to locate items of interest, it is usually easier and more efficient to get some form of human assistance first, and, with few exceptions, this request for assistance is presented in the form of a oral query. In 60 addition, many persons cannot or will not, because of physical or psychological barriers, use any of the aforementioned conventional I/O devices. For example, many older persons cannot easily read the text presented on WWW pages, or understand the layout/hierarchy of menus, or 65 manipulate a mouse to make finely coordinated movements to indicate their selections. Many others are intimidated by

2

the look and complexity of computer systems, WWW pages, etc., and therefore do not attempt to use online services for this reason as well.

Thus, applications which can mimic normal human interactions are likely to be preferred by potential on-line shoppers and persons looking for information over the WWW. It is also expected that the use of voice-based systems will increase the universe of persons willing to engage in e-commerce, e-learning, etc. To date, however, there are very few systems, if any, which permit this type of interaction, and, if they do, it is very limited. For example, various commercial programs sold by IBM (VIAVOICETM) and Kurzweil (DRAGONTM) permit some user control of the interface (opening, closing files) and searching (by using previously trained URLs) but they do not present a flexible solution that can be used by a number of users across multiple cultures and without time consuming voice training. Typical prior efforts to implement voice based functionality in an INTER-NET context can be seen in U.S. Pat. No. 5,819,220 incorporated by reference herein.

Another issue presented by the lack of voice-based systems is efficiency. Many companies are now offering technical support over the INTERNET, and some even offer live operator assistance for such queries. While this is very advantageous (for the reasons mentioned above) it is also extremely costly and inefficient, because a real person must be employed to handle such queries. This presents a practical limit that results in long wait times for responses or high labor overheads. An example of this approach can be seen U.S. Pat. No. 5,802,526 also incorporated by reference herein. In general, a service presented over the WWW is far more desirable if it is "scalable," or, in other words, able to handle an increasing amount of user traffic with little if any perceived delay or troubles by a prospective user.

In a similar context, while remote learning has become an increasingly popular option for many students, it is practically impossible for an instructor to be able to field questions from more than one person at a time. Even then, such interaction usually takes place for only a limited period of time because of other instructor time constraints. To date, however, there is no practical way for students to continue a human-like question and answer type dialog after the learning session is over, or without the presence of the instructor to personally address such queries.

Conversely, another aspect of emulating a human-like dialog involves the use of oral feedback. In other words, many persons prefer to receive answers and information in audible form. While a form of this functionality is used by some websites to communicate information to visitors, it is not performed in a real-time, interactive question-answer dialog fashion so its effectiveness and usefulness is limited.

Yet another area that could benefit from speech-based interaction involves so-called "search" engines used by INTERNET users to locate information of interest at web sites, such as the those available at YAHOO®.com, METACRAWLER®.com, EXCITE®.com, etc. These tools permit the user to form a search query using either combinations of keywords or metacategories to search through a web page database containing text indices associated with one or more distinct web pages. After processing the user's request, therefore, the search engine returns a number of hits which correspond, generally, to URL pointers and text excerpts from the web pages that represent the closest match made by such search engine for the particular user query based on the search processing logic used by search engine. The structure and operation of such prior art search engines, including the mechanism by which they build the web page

3

database, and parse the search query, are well known in the art. To date, applicant is unaware of any such search engine that can easily and reliably search and retrieve information based on speech input from a user.

There are a number of reasons why the above environ- 5 ments (e-commerce, e-support, remote learning, INTER-NET searching, etc.) do not utilize speech-based interfaces, despite the many benefits that would otherwise flow from such capability. First, there is obviously a requirement that the output of the speech recognizer be as accurate as possible. One of the more reliable approaches to speech recognition used at this time is based on the Hidden Markov Model (HMM)—a model used to mathematically describe any time series. A conventional usage of this technique is disclosed, for example, in U.S. Pat. No. 4,587,670 incorpo- 15 rated by reference herein. Because speech is considered to have an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated 20 with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. This finite state 25 machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_i(O_t)$. It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name 30 Hidden Markov Model. The basic theory of HMMs was published in a series of classic papers by Baum and his colleagues in the late 1960's and early 1970's. HMMs were first used in speech applications by Baker at Carnegie Mellon, by Jelenik and colleagues at IBM in the late 1970's 35 and by Steve Young and colleagues at Cambridge University, UK in the 1990's. Some typical papers and texts are as

- L. E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains", Ann. 40 Math. Stat., 37: 1554-1563, 1966
- L. E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1-8, 1972
- J. H. Baker, "The dragon system—An Overview", IEEE Trans. on ASSP Proc., ASSP-23(1): 24-29, February 1975
- F. Jeninek et al, "Continuous Speech Recognition: Statistical methods" in Handbook of Statistics, II, P. R. 50 Kristnaiad, Ed. Amsterdam, The Netherlands, North-Holland, 1982
- L. R. Bahl, F. Jeninek, R. L. Mercer, "A maximum likelihood approach to continuous speech recognition", IEEE Trans. Pattern Anal. Mach. Intell., PAMI-5: 179-55 190.1983
- J. D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, N.J. 1980.
- H. R. Rabiner and B. H. Juang, "Fundamentals of 60 Speech Recognition", Prentice Hall, 1993
- H. R. Rabiner, "Digital Processing of Speech Signals", Prentice Hall, 1978

More recently research has progressed in extending HMM and combining HMMs with neural networks to speech 65 recognition applications at various laboratories. The following is a representative paper:

4

9. Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco (1993), Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition. Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4 pp. 899-916. Also in I. Guyon and P. Wang editors, Advances in Pattern Recognition Systems using Neural Networks, Vol. 7 of a Series in Machine Perception and Artificial Intelligence. World Scientific, February 1994.

All of the above are hereby incorporated by reference. While the HMM-based speech recognition yields very good results, contemporary variations of this technique cannot guarantee a word accuracy requirement of 100% exactly and consistently, as will be required for WWW applications for all possible all user and environment conditions. Thus, although speech recognition technology has been available for several years, and has improved significantly, the technical requirements have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily.

In contrast to word recognition, Natural language processing (NLP) is concerned with the parsing, understanding and indexing of transcribed utterances and larger linguistic units. Because spontaneous speech contains many surface phenomena such as disfluencies,-hesitations, repairs and restarts, discourse markers such as 'well' and other elements which cannot be handled by the typical speech recognizer, it is the problem and the source of the large gap that separates speech recognition and natural language processing technologies. Except for silence between utterances, another problem is the absence of any marked punctuation available for segmenting the speech input into meaningful units such as utterances. For optimal NLP performance, these types of phenomena should be annotated at its input. However, most continuous speech recognition systems produce only a raw sequence of words. Examples of conventional systems using NLP are shown in U.S. Pat. Nos. 4,991,094, 5,068,789, 5,146,405 and 5,680,628, all of which are incorporated by reference herein.

Second, most of the very reliable voice recognition systems are speaker-dependent, requiring that the interface be "trained" with the user's voice, which takes a lot of time, and is thus very undesirable from the perspective of a WWW environment, where a user may interact only a few times with a particular website. Furthermore, speaker-dependent systems usually require a large user dictionary (one for each unique user) which reduces the speed of recognition. This makes it much harder to implement a real-time dialog interface with satisfactory response capability (i.e., something that mirrors normal conversation—on the order of 3-5 seconds is probably ideal). At present, the typical shrinkwrapped speech recognition application software include offerings from IBM (VIAVOICETM) and Dragon Systems (DRAGONTM). While most of these applications are adequate for dictation and other transcribing applications, they are woefully inadequate for applications such as NLQS where the word error rate must be close to 0%. In addition these offerings require long training times and are typically are non client-server configurations. Other types of trained systems are discussed in U.S. Pat. No. 5,231,670 assigned to Kurzweil, and which is also incorporated by reference

Another significant problem faced in a distributed voicebased system is a lack of uniformity/control in the speech recognition process. In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on

5

a single client. A well-known system of this type is depicted in U.S. Pat. No. 4,991,217 incorporated by reference herein. These clients can take numerous forms (desktop PC, laptop PC, PDA, etc.) having varying speech signal processing and communications capability. Thus, from the server side perspective, it is not easy to assure uniform treatment of all users accessing a voice-enabled web page, since such users may have significantly disparate word recognition and error rate performances. While a prior art reference to Gould et al.—U.S. Pat. No. 5,915,236—discusses generally the notion of tailoring a recognition process to a set of available computational resources, it does not address or attempt to solve the issue of how to optimize resources in a distributed environment such as a client-server model. Again, to enable such voice-based technologies on a wide-spread scale it is far more preferable to have a system that harmonizes and accounts for discrepancies in individual systems so that even the thinnest client is supportable, and so that all users are able to interact in a satisfactory manner with the remote server running the e-commerce, e-support and/or remote 20 learning application.

Two references that refer to a distributed approach for speech recognition include U.S. Pat. Nos. 5,956,683 and 5,960,399 incorporated by reference herein. In the first of these, U.S. Pat. No. 5,956,683—Distributed Voice Recognition System (assigned to Qualcomm) an implementation of a distributed voice recognition system between a telephony-based handset and a remote station is described. In this implementation, all of the word recognition operations seem to take place at the handset. This is done since the patent describes the benefits that result from locating of the system for acoustic feature extraction at the portable or cellular phone in order to limit degradation of the acoustic features due to quantization distortion resulting from the narrow bandwidth telephony channel. This reference therefore does not address the issue of how to ensure adequate performance for a very thin client platform. Moreover, it is difficult to determine, how, if at all, the system can perform real-time word recognition, and there is no meaningful description of how to integrate the system with a natural language processor.

The second of these references—U.S. Pat. No. 5,960, 399—Client/Server Speech Processor/Recognizer (assigned to GTE) describes the implementation of a HMM-based 45 distributed speech recognition system. This reference is not instructive in many respects, however, including how to optimize acoustic feature extraction for a variety of client platforms, such as by performing a partial word recognition process where appropriate. Most importantly, there is only a $_{50}$ description of a primitive server-based recognizer that only recognizes the user's speech and simply returns certain keywords such as the user's name and travel destination to fill out a dedicated form on the user's machine. Also, the streaming of the acoustic parameters does not appear to be 55 implemented in real-time as it can only take place after silence is detected. Finally, while the reference mentions the possible use of natural language processing (column 9) there is no explanation of how such function might be implemented in a real-time fashion to provide an interactive feel for the user.

SUMMARY OF THE INVENTION

An object of the present invention, therefore, is to provide 65 an improved system and method for overcoming the limitations of the prior art noted above;

6

A primary object of the present invention is to provide a word and phrase recognition system that is flexibly and optimally distributed across a client/platform computing architecture, so that improved accuracy, speed and uniformity can be achieved for a wide group of users;

A further object of the present invention is to provide a speech recognition system that efficiently integrates a distributed word recognition system with a natural language processing system, so that both individual words and entire speech utterances can be quickly and accurately recognized in any number of possible languages;

A related object of the present invention is to provide an efficient query response system so that an extremely accurate, real-time set of appropriate answers can be given in response to speech-based queries;

Yet another object of the present invention is to provide an interactive, real-time instructional/learning system that is distributed across a client/server architecture, and permits a real-time question/answer session with an interactive character:

A related object of the present invention is to implement such interactive character with an articulated response capability so that the user experiences a human-like interaction;

Still a further object of the present invention is to provide an INTERNET website with speech processing capability so that voice based data and commands can be used to interact with such site, thus enabling voice-based e-commerce and e-support services to be easily scaleable;

Another object is to implement a distributed speech recognition system that utilizes environmental variables as part of the recognition process to improve accuracy and speed;

A further object is to provide a scaleable query/response database system, to support any number of query topics and users as needed for a particular application and instantaneous demand;

Yet another object of the present invention is to provide a query recognition system that employs a two-step approach, including a relatively rapid first step to narrow down the list of potential responses to a smaller candidate set, and a second more computationally intensive second step to identify the best choice to be returned in response to the query from the candidate set:

A further object of the present invention is to provide a natural language processing system that facilitates query recognition by extracting lexical components of speech utterances, which components can be used for rapidly identifying a candidate set of potential responses appropriate for such speech utterances;

Another related object of the present invention is to provide a natural language processing system that facilitates query recognition by comparing lexical components of speech utterances with a candidate set of potential response to provide an extremely accurate best response to such query.

One general aspect of the present invention, therefore, relates to a natural language query system (NLQS) that offers a fully interactive method for answering user's questions over a distributed network such as the INTERNET or a local intranet. This interactive system when implemented over the worldwide web (WWW) services of the INTERNET functions so that a client or user can ask a question in a natural language such as English, French, German or Spanish and receive the appropriate answer at his or her personal computer also in his or her native natural language.

The system is distributed and consists of a set of integrated software modules at the client's machine and another

7

set of integrated software programs resident on a server or set of servers. The client-side software program is comprised of a speech recognition program, an agent and its control program, and a communication program. The server-side program is comprised of a communication program, a natural language engine (NLE), a database processor (DBProcess), an interface program for interfacing the DBProcess with the NLE, and a SQL database. In addition, the client's machine is equipped with a microphone and a speaker. Processing of the speech utterance is divided between the client and server side so as to optimize processing and transmission latencies, and so as to provide support for even very thin client platforms.

In the context of an interactive learning application, the system is specifically used to provide a single-best answer to a user's question. The question that is asked at the client's machine is articulated by the speaker and captured by a microphone that is built in as in the case of a notebook computer or is supplied as a standard peripheral attachment. 20 Once the question is captured, the question is processed partially by NLQS client-side software resident in the client's machine. The output of this partial processing is a set of speech vectors that are transported to the server via the INTERNET to complete the recognition of the user's questions. This recognized speech is then converted to text at the server.

After the user's question is decoded by the speech recognition engine (SRE) located at the server, the question is converted to a structured query language (SQL) query. This query is then simultaneously presented to a software process within the server called DBProcess for preliminary processing and to a Natural Language Engine (NLE) module for extracting the noun phrases (NP) of the user's question. During the process of extracting the noun phrase within the NLE, the tokens of the users' question are tagged. The tagged tokens are then grouped so that the NP list can be determined. This information is stored and sent to the DBProcess process.

In the DBProcess, the SQL query is fully customized using the NP extracted from the user's question and other environment variables that are relevant to the application. For example, in a training application, the user's selection of course, chapter and or section would constitute the environment variables. The SQL query is constructed using the extended SQL Full-Text predicates—CONTAINS, FREET-EXT, NEAR, AND. The SQL query is next sent to the Full-Text search engine within the SQL database, where a Full-Text search procedure is initiated. The result of this search procedure is recordset of answers. This recordset contains stored questions that are similar linguistically to the user's question. Each of these stored questions has a paired answer stored in a separate text file, whose path is stored in a table of the database.

The entire recordset of returned stored answers is then returned to the NLE engine in the form of an array. Each stored question of the array is then linguistically processed sequentially one by one. This linguistic processing constitutes the second step of a 2-step algorithm to determine the 60 single best answer to the user's question. This second step proceeds as follows: for each stored question that is returned in the recordset, a NP of the stored question is compared with the NP of the user's question. After all stored questions of the array are compared with the user's question, the stored question that yields the maximum match with the user's question is selected as the best possible stored question that

matches the user's question. The metric that is used to determine the best possible stored question is the number of

The stored answer that is paired to the best-stored question is selected as the one that answers the user's question. The ID tag of the question is then passed to the DBProcess. This DBProcess returns the answer which is stored in a file.

A communication link is again established to send the answer back to the client in compressed form. The answer once received by the client is decompressed and articulated to the user by the text-to-speech engine. Thus, the invention can be used in any number of different applications involving interactive learning systems, INTERNET related commerce sites, INTERNET search engines, etc.

Computer-assisted instruction environments often require the assistance of mentors or live teachers to answer questions from students. This assistance often takes the form of organizing a separate pre-arranged forum or meeting time that is set aside for chat sessions or live call-in sessions so that at a scheduled time answers to questions may be provided. Because of the time immediacy and the ondemand or asynchronous nature of on-line training where a student may log on and take instruction at any time and at any location, it is important that answers to questions be provided in a timely and cost-effective manner so that the user or student can derive the maximum benefit from the material presented.

This invention addresses the above issues. It provides the user or student with answers to questions that are normally channeled to a live teacher or mentor. This invention provides a single-best answer to questions asked by the student. The student asks the question in his or her own voice in the language of choice. The speech is recognized and the answer to the question is found using a number of technologies including distributed speech recognition, full-text search database processing, natural language processing and textto-speech technologies. The answer is presented to the user, as in the case of a live teacher, in an articulated manner by an agent that mimics the mentor or teacher, and in the language of choice—English, French, German, Japanese or other natural spoken language. The user can choose the agent's gender as well as several speech parameters such as pitch, volume and speed of the character's voice.

Other applications that benefit from NLQS are e-commerce applications. In this application, the user's query for a price of a book, compact disk or for the availability of any item that is to be purchased can be retrieved without the need to pick through various lists on successive web pages. Instead, the answer is provided directly to the user without any additional user input.

Similarly, it is envisioned that this system can be used to provide answers to frequently-asked questions (FAQs), and as a diagnostic service tool for e-support. These questions are typical of a give web site and are provided to help the user find information related to a payment procedure or the specifications of, or problems experienced with a product/service. In all of these applications, the NLQS architecture can be applied.

A number of inventive methods associated with these architectures are also beneficially used in a variety of INTERNET related applications.

Although the inventions are described below in a set of preferred embodiments, it will be apparent to those skilled in the art the present inventions could be beneficially used in many environments where it is necessary to implement fast, accurate speech recognition, and/or to provide a human-like dialog capability to an intelligent system.

q

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a preferred embodiment of a natural language query system (NLQS) of the present invention, which is distributed across a client/server computing architecture, and can be used as an interactive learning system, an e-commerce system, an e-support system, and the like;

FIGS. **2**A-**2**C are a block diagram of a preferred embodiment of a client side system, including speech capturing ¹⁰ modules, partial speech processing modules, encoding modules, transmission modules, agent control modules, and answer/voice feedback modules that can be used in the aforementioned NLQS;

FIG. 2D is a block diagram of a preferred embodiment of ¹⁵ a set of initialization routines and procedures used for the client side system of FIGS. 2A-2C;

FIG. 3 is a block diagram of a preferred embodiment of a set of routines and procedures used for handling an iterated set of speech utterances on the client side system of FIGS. ²⁰ **2A-2C**, transmitting speech data for such utterances to a remote server, and receiving appropriate responses back from such server;

FIG. 4 is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for ²⁵ un-initializing the client side system of FIGS. 2A-2C;

FIG. 4A is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a distributed component of a speech recognition module for the server side system of FIG. 5;

FIG. 4B is a block diagram of a preferred set of routines and procedures used for implementing an SQL query builder for the server side system of FIG. 5;

FIG. 4C is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a database control process module for the server side system of FIG. 5;

FIG. 4D is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a natural language engine that provides query formulation support, a query response module, and an interface to the database control process module for the server side system of FIG. 5:

FIG. 5 is a block diagram of a preferred embodiment of a server side system, including a speech recognition module to complete processing of the speech utterances, environmental and grammar control modules, query formulation modules, a natural language engine, a database control module, and a query response module that can be used in the aforementioned NLQS;

FIG. 6 illustrates the organization of a full-text database used as part of server side system shown in FIG. 5;

FIG. 7A illustrates the organization of a full-text database course table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention:

FIG. 7B illustrates the organization of a full-text database chapter table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention:

FIG. 7C describes the fields used in a chapter table used as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention;

FIG. 7D describes the fields used in a section table used 65 as part of server side system shown in FIG. 5 for an interactive learning embodiment of the present invention;

10

FIG. **8** is a flow diagram of a first set of operations performed by a preferred embodiment of a natural language engine on a speech utterance including Tokenization, Tagging and Grouping:

FIG. 9 is a flow diagram of the operations performed by a preferred embodiment of a natural language engine on a speech utterance including stemming and Lexical Analysis

FIG. 10 is a block diagram of a preferred embodiment of a SQL database search and support system for the present invention:

FIGS. 11A-11C are flow diagrams illustrating steps performed in a preferred two step process implemented for query recognition by the NLQS of FIG. 2;

FIG. 12 is an illustration of another embodiment of the present invention implemented as part of a Web-based speech based learning/training System;

FIGS. 13-17 are illustrations of another embodiment of the present invention implemented as part of a Web-based e-commerce system;

FIG. 18 is an illustration of another embodiment of the present invention implemented as part of a voice-based Help Page for an E-Commerce Web Site.

DETAILED DESCRIPTION OF THE INVENTION

Overview

As alluded to above, the present inventions allow a user to ask a question in a natural language such as English, French, German, Spanish or Japanese at a client computing system (which can be as simple as a personal digital assistant or cell-phone, or as sophisticated as a high end desktop PC) and receive an appropriate answer from a remote server also in his or her native natural language. As such, the embodiment of the invention shown in FIG. 1 is beneficially used in what can be generally described as a Natural Language Query System (NLQS) 100, which is configured to interact on a real-time basis to give a human-like dialog capability/experience for e-commerce, e-support, and e-learning applications

The processing for NLQS 100 is generally distributed across a client side system 150, a data link 160, and a server-side system 180. These components are well known in the art, and in a preferred embodiment include a personal computer system 150, an INTERNET connection 160A, 160B, and a larger scale computing system 180. It will be understood by those skilled in the art that these are merely exemplary components, and that the present invention is by no means limited to any particular implementation or combination of such systems. For example, client-side system 150 could also be implemented as a computer peripheral, a PDA, as part of a cell-phone, as part of an INTERNETadapted appliance, an INTERNET linked kiosk, etc. Similarly, while an INTERNET connection is depicted for data link 160A, it is apparent that any channel that is suitable for carrying data between client system 150 and server system 180 will suffice, including a wireless link, an RF link, an IR link, a LAN, and the like. Finally, it will be further appreciated that server system 180 may be a single, large-scale system, or a collection of smaller systems interlinked to support a number of potential network users.

Initially speech input is provided in the form of a question or query articulated by the speaker at the client's machine or personal accessory as a speech utterance. This speech utterance is captured and partially processed by NLQS clientside software 155 resident in the client's machine. To

11

facilitate and enhance the human-like aspects of the interaction, the question is presented in the presence of an animated character 157 visible to the user who assists the user as a personal information retriever/agent. The agent can also interact with the user using both visible text output on 5 a monitor/display (not shown) and/or in audible form using a text to speech engine 159. The output of the partial processing done by SRE 155 is a set of speech vectors that are transmitted over communication channel 160 that links the user's machine or personal accessory to a server or 10 servers via the INTERNET or a wireless gateway that is linked to the INTERNET as explained above. At server 180, the partially processed speech signal data is handled by a server-side SRE 182, which then outputs recognized speech text corresponding to the user's question. Based on this user 15 question related text, a text-to-query converter 184 formulates a suitable query that is used as input to a database processor 186. Based on the query, database processor 186 then locates and retrieves an appropriate answer using a customized SQL query from database 188. A Natural Lan- 20 guage Engine 190 facilitates structuring the query to database 188. After a matching answer to the user's question is found, the former is transmitted in text form across data link 160B, where it is converted into speech by text to speech engine 159, and thus expressed as oral feedback by animated 25 character agent 157.

Because the speech processing is broken up in this fashion, it is possible to achieve real-time, interactive, human-like dialog consisting of a large, controllable set of questions/answers. The assistance of the animated agent 157 30 further enhances the experience, making it more natural and comfortable for even novice users. To make the speech recognition process more reliable, context-specific grammars and dictionaries are used, as well as natural language processing routines at NLE 190, to analyze user questions 35 lexically. While context-specific processing of speech data is known in the art (see e.g., U.S. Pat. Nos. 5,960,394, 5,867, 817, 5,758,322 and 5,384,892 incorporated by reference herein) the present inventors are unaware of any such implementation as embodied in the present inventions. The 40 text of the user's question is compared against text of other questions to identify the question posed by the user by DB processor/engine (DBE) 186. By optimizing the interaction and relationship of the SR engines 155 and 182, the NLP routines 190, and the dictionaries and grammars, an 45 extremely fast and accurate match can be made, so that a unique and responsive answer can be provided to the user.

On the server side 180, interleaved processing further accelerates the speech recognition process. In simplified terms, the query is presented simultaneously both to NLE 50 190 after the query is formulated, as well as to DBE 186. NLE 190 and SRE 182 perform complementary functions in the overall recognition process. In general, SRE 182 is primarily responsible for determining the identity of the words articulated by the user, while NLE 190 is responsible 55 for the linguistic morphological analysis of both the user's query and the search results returned after the database query.

After the user's query is analyzed by NLE 190 some parameters are extracted and sent to the DBProcess. Additional statistics are stored in an array for the 2^{nd} step of processing. During the 2^{nd} step of 2-step algorithm, the recordset of preliminary search results are sent to the NLE 160 for processing. At the end of this 2^{nd} step, the single question that matches the user's query is sent to the DBProcess where further processing yields the paired answer that is paired with the single best stored question.

12

Thus, the present invention uses a form of natural language processing (NLP) to achieve optimal performance in a speech based web application system. While NLP is known in the art, prior efforts in Natural Language Processing (NLP) work nonetheless have not been well integrated with Speech Recognition (SR) technologies to achieve reasonable results in a web-based application environment. In speech recognition, the result is typically a lattice of possible recognized words each with some probability of fit with the speech recognizer. As described before, the input to a typical NLP system is typically a large linguistic unit. The NLP system is then charged with the parsing, understanding and indexing of this large linguistic unit or set of transcribed utterances. The result of this NLP process is to understand lexically or morphologically the entire linguistic unit as opposed to word recognition. Put another way, the linguistic unit or sentence of connected words output by the SRE has to be understood lexically, as opposed to just being "recog-

As indicated earlier, although speech recognition technology has been available for several years, the technical requirements for the NLQS invention have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily. In realizing that even with the best of conditions, it might be not be possible to achieve the perfect 100% speech recognition accuracy that is required, the present invention employs an algorithm that balances the potential risk of the speech recognition process with the requirements of the natural language processing so that even in cases where perfect speech recognition accuracy is not achieved for each word in the query, the entire query itself is none-theless recognized with sufficient accuracy.

This recognition accuracy is achieved even while meeting very stringent user constraints, such as short latency periods of 3 to 5 seconds (ideally—ignoring transmission latencies which can vary) for responding to a speech-based query, and for a potential set of 100-250 query questions. This quick response time gives the overall appearance and experience of a real-time discourse that is more natural and pleasant from the user's perspective. Of course, non-real time applications, such as translation services for example, can also benefit from the present teachings as well, since a centralized set of HMMs, grammars, dictionaries, etc., are maintained.

General Aspects of Speech Recognition used in the Present Inventions

General background information on speech recognition can be found in the prior art references discussed above and incorporated by reference herein. Nonetheless, a discussion of some particular exemplary forms of speech recognition structures and techniques that are well-suited for NLQS 100 is provided next to better illustrate some of the characteristics, qualities and features of the present inventions.

Speech recognition technology is typically of two types—speaker independent and speaker dependent. In speaker-dependent speech recognition technology, each user has a voice file in which a sample of each potentially recognized word is stored. Speaker-dependent speech recognition systems typically have large vocabularies and dictionaries making them suitable for applications as dictation and text transcribing. It follows also that the memory and processor resource requirements for the speaker-dependent can be and are typically large and intensive.

13

Conversely speaker-independent speech recognition technology allows a large group of users to use a single vocabulary file. It follows then that the degree of accuracy that can be achieved is a function of the size and complexity of the grammars and dictionaries that can be supported for a given language. Given the context of applications for which NLQS, the use of small grammars and dictionaries allow speaker independent speech recognition technology to be implemented in NLQS.

The key issues or requirements for either type—speakerindependent or speaker-dependent, are accuracy and speed. As the size of the user dictionaries increase, the speech recognition accuracy metric-word error rate (WER) and the speed of recognition decreases. This is so because the 15 search time increases and the pronunciation match becomes more complex as the size of the dictionary increases.

The basis of the NLQS speech recognition system is a series of Hidden Markov Models (HMM), which, as alluded to earlier, are mathematical models used to characterize any time varying signal. Because parts of speech are considered to be based on an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to an associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O, is generated with probability density B₂(O₂). It is only the outcome, not the state which is visible to an external 35 observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.

In isolated speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word can each be described by a Markov model as follows: 40

$$O=o_1, o_2, \dots o_T \tag{1-1}$$

where o, is a speech vector observed at time t. The isolated word recognition then is to compute:

$$\arg\max\left\{P(\mathbf{w}_{i}|\mathbf{O})\right\} \tag{1-2}$$

By using Bayes' Rule.

$${P(w_i|O)}=[P(O|w_i)P(w_i)]/P(O)$$
 (1-3)

In the general case, the Markov model when applied to speech also assumes a finite state machine which changes state once every time unit and each time that a state j is entered, a speech vector o, is generated from the probability density b_i (o_t). Furthermore, the transition from state i to 55 state j is also probabilistic and is governed by the discrete probability a_{ii}.

For a state sequence X, the joint probability that O is generated by the model M moving through a state sequence X is the product of the transition probabilities and the output 60 probabilities. Only the observation sequence is known—the state sequence is hidden as mentioned before.

Given that X is unknown, the required likelihood is computed by summing over all possible state sequences $X=x(1), x(2), x(3) \dots x(T)$, that is

$$P(O|M) = \Sigma \big\{ a_{x(0)x(1)} \Pi b(x)(o_t) a_{x(t)x(t+1)} \big\}$$

14

Given a set of models M_i, corresponding to words w_i equation 1-2 is solved by using 1-3 and also by assuming

$$P(O|w_i)=P(O|M_i)$$

All of this assumes that the parameters $\{a_{ij}\}$ and $\{b_i(o_i)\}$ are known for each model M_i. This can be done, as explained earlier, by using a set of training examples corresponding to a particular model. Thereafter, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure. So if a sufficient number of representative examples of each word are collected, then a HMM can be constructed which simply models all of the many sources of variability inherent in real speech. This training is well-known in the art, so it is not described at length herein. except to note that the distributed architecture of the present invention enhances the quality of HMMs, since they are derived and constituted at the server side, rather than the client side. In this way, appropriate samples from users of different geographical areas can be easily compiled and analyzed to optimize the possible variations expected to be seen across a particular language to be recognized. Uniformity of the speech recognition process is also well-maintained, and error diagnostics are simplified, since each prospective user is using the same set of HMMs during the recognition process.

To determine the parameters of a HMM from a set of training samples, the first step typically is to make a rough guess as to what they might be. Then a refinement is done using the Baum-Welch estimation formulae. By these formulae, the maximum likelihood estimates of μ_i (where μ_i is mean vector and Σ_i is covariance matrix) is:

$$\mu_{i} = \sum_{t=1}^{T} L_{i}(t)o_{t}/[\sum_{t=1}^{T} L_{i}(t)o_{t}]$$

A forward-backward algorithm is next used to calculate the probability of state occupation L_i(t). If the forward probability $\alpha_i(t)$ for some model M with N states is defined

$$\alpha_i(t)=P(o_1,\ldots,o_nx(t)=jM)$$

This probability can be calculated using the recursion:

$$\alpha_j(t) = [\Sigma^{N-1}_{i=2}\alpha(t-1)a_{ij}]b_j(o_t)$$

Similarly the backward probability can be computed using the recursion:

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij} b_j(o_{t+1})(t+1)$$

Realizing that the forward probability is a joint probability and the backward probability is a conditional probability, the probability of state occupation is the product of the two probabilities:

$$\alpha j(t)\beta_i(t)=P(O,x(t)=j|M)$$

Hence the probability of being in state i at a time t is:

$$L_j(t)=1/P[\alpha_j(t)\beta_j(t)]$$

where P=P(O|M)

To generalize the above for continuous speech recognition, we assume the maximum likelihood state sequence where the summation is replaced by a maximum operation. Thus for a given model M, let ϕj (t) represent the maximum likelihood of observing speech vectors o, to o, and being used in state j at time t:

$$\phi_i(t) = \max\{\phi_i(t)(t-1)\alpha_{ii}\}\beta_i(o_t)$$

Expressing this logarithmically to avoid underflow, this likelihood becomes:

$$\psi_i(t) = \max\{\psi_i(t-1) + \log(\alpha_{ii})\} + \log(b_i(o_t))$$

This is also known as the Viterbi algorithm. It can be visualized as finding the best path through a matrix where the vertical dimension represents the states of the HMM and

15

horizontal dimension represents frames of speech i.e. time. To complete the extension to connected speech recognition, it is further assumed that each HMM representing the underlying sequence is connected. Thus the training data for continuous speech recognition should consist of connected outterances; however, the boundaries between words do not have to be known.

To improve computational speed/efficiency, the Viterbi algorithm is sometimes extended to achieve convergence by using what is known as a Token Passing Model. The token 10 passing model represents a partial match between the observation sequence o₁ to o₂ and a particular model, subject to the constraint that the model is in state j at time t. This token passing model can be extended easily to connected speech environments as well if we allow the sequence of HMMs to 15 be defined as a finite state network. A composite network that includes both phoneme-based HMMs and complete words can be constructed so that a single-best word can be recognized to form connected speech using word N-best extraction from the lattice of possibilities. This composite 20 form of HMM-based connected speech recognizer is the basis of the NLQS speech recognizer module. Nonetheless, the present invention is not limited as such to such specific forms of speech recognizers, and can employ other techniques for speech recognition if they are otherwise compat- 25 ible with the present architecture and meet necessary performance criteria for accuracy and speed to provide a real-time dialog experience for users.

The representation of speech for the present invention's HMM-based speech recognition system assumes that speech 30 is essentially either a quasi-periodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced sounds). It may be modeled as two sources—one a impulse train generator with pitch period P and a random noise generator which is controlled by a voice/unvoiced switch. 35 The output of the switch is then fed into a gain function estimated from the speech signal and scaled to feed a digital filter H(z) controlled by the vocal tract parameter characteristics of the speech being produced. All of the parameters for this model—the voiced/unvoiced switching, the pitch 40 period for voiced sounds, the gain parameter for the speech signal and the coefficient of the digital filter, vary slowly with time. In extracting the acoustic parameters from the user's speech input so that it can evaluated in light of a set of HMMs, cepstral analysis is typically used to separate the 45 vocal tract information from the excitation information. The cepstrum of a signal is computed by taking the Fourier (or similar) transform of the log spectrum. The principal advantage of extracting cepstral coefficients is that they are de-correlated and the diagonal covariances can be used with 50 HMMs. Since the human ear resolves frequencies nonlinearly across the audio spectrum, it has been shown that a front-end that operates in a similar non-linear way improves speech recognition performance.

Accordingly, instead of a typical linear prediction-based 55 analysis, the front-end of the NLQS speech recognition engine implements a simple, fast Fourier transform based filter bank designed to give approximately equal resolution on the Mel-scale. To implement this filter bank, a window of speech data (for a particular time frame) is transformed 60 using a software based Fourier transform and the magnitude taken. Each FFT magnitude is then multiplied by the corresponding filter gain and the results accumulated. The cepstral coefficients that are derived from this filter-bank analysis at the front end are calculated during a first partial 65 processing phase of the speech signal by using a Discrete Cosine Transform of the log filter bank amplitudes. These

16

cepstral coefficients are called Mel-Frequency Cepstral Coefficients (MFCC) and they represent some of the speech parameters transferred from the client side to characterize the acoustic features of the user's speech signal. These parameters are chosen for a number of reasons, including the fact that they can be quickly and consistently derived even across systems of disparate capabilities (i.e., for everything from a low power PDA to a high powered desktop system), they give good discrimination, they lend themselves to a number of useful recognition related manipulations, and they are relatively small and compact in size so that they can be transported rapidly across even a relatively narrow band link. Thus, these parameters represent the least amount of information that can be used by a subsequent server side system to adequately and quickly complete the recognition process.

To augment the speech parameters an energy term in the form of the logarithm of the signal energy is added. Accordingly, RMS energy is added to the 12 MFCC's to make 13 coefficients. These coefficients together make up the partially processed speech data transmitted in compressed form from the user's client system to the remote server side.

The performance of the present speech recognition system is enhanced significantly by computing and adding time derivatives to the basic static MFCC parameters at the server side. These two other sets of coefficients—the delta and acceleration coefficients representing change in each of the 13 values from frame to frame (actually measured across several frames), are computed during a second partial speech signal processing phase to complete the initial processing of the speech signal, and are added to the original set of coefficients after the latter are received. These MFCCs together with the delta and acceleration coefficients constitute the observation vector O, mentioned above that is used for determining the appropriate HMM for the speech data.

The delta and acceleration coefficients are computed using the following regression formula:

$$d_t \!\!=\!\! \boldsymbol{\Sigma}^{\boldsymbol{\theta}}_{\phantom{\boldsymbol{\theta}} = 1} [c_{t+\boldsymbol{\theta}} \!\!-\!\! c_{t-\boldsymbol{\theta}}] / 2 \boldsymbol{\Sigma}^{\boldsymbol{\theta}}_{\phantom{\boldsymbol{\theta}} = 1} \boldsymbol{\theta}^2$$

where d, is a delta coefficient at time t computed in terms of the corresponding static coefficients:

$$d_t = [c_{t+\theta} - c_{t-\theta}]/2\theta$$

In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. In other words, both the first and second partial processing phases above are executed by the same DSP (or microprocessor) running a ROM or software code routine at the client's computing machine.

In contrast, because of several considerations, specifically-cost, technical performance, and client hardware uniformity, the present NLQS system uses a partitioned or distributed approach. While some processing occurs on the client side, the main speech recognition engine runs on a centrally located server or number of servers. More specifically, as noted earlier, capture of the speech signals, MFCC vector extraction and compression are implemented on the client's machine during a first partial processing phase. The routine is thus streamlined and simple enough to be implemented within a browser program (as a plug in module, or a downloadable applet for example) for maximum ease of use and utility. Accordingly, even very "thin" client platforms can be supported, which enables the use of the present system across a greater number of potential sites. The primary MFCCs are then transmitted to the server over the channel, which, for example, can include a dial-up INTER-

17

NET connection, a LAN connection, a wireless connection and the like. After decompression, the delta and acceleration coefficients are computed at the server to complete the initial speech processing phase, and the resulting observation vectors O, are also determined.

General Aspects of Speech Recognition Engine

The speech recognition engine is also located on the server, and is based on a HTK-based recognition network compiled from a word-level network, a dictionary and a set of HMMs. The recognition network consists of a set of nodes connected by arcs. Each node is either a HMM model instance or a word end. Each model node is itself a network consisting of states connected by arcs. Thus when fully compiled, a speech recognition network consists of HMM states connected by transitions. For an unknown input utterance with T frames, every path from the start node to the exit node of the network passes through T HMM states. Each of these paths has log probability which is computed by summing the log probability of each individual transition in the path and the log probability of each emitting state generating the corresponding observation. The function of the Viterbi decoder is find those paths through the network which have the highest log probability. This is found using the Token Passing algorithm. In a network that has many nodes, the computation time is reduced by only allowing propagation of those tokens which will have some chance of becoming winners. This process is called pruning.

Natural Language Processor

In a typical natural language interface to a database, the 30 user enters a question in his/her natural language, for example, English. The system parses it and translates it to a query language expression. The system then uses the query language expression to process the query and if the search is successful, a recordset representing the results is displayed 35 in English either formatted as raw text or in a graphical form. For a natural language interface to work well involves a number of technical requirements.

For example, it needs to be robust—in the sentence 'What's the departments turnover' it needs to decide that the 40 word whats=what's=what is. And it also has to determine that departments=department's. In addition to being robust, the natural language interface has to distinguish between the several possible forms of ambiguity that may exist in the natural language—lexical, structural, reference and ellipsis 45 ambiguity. All of these requirements, in addition to the general ability to perform basic linguistic morphological operations of tokenization, tagging and grouping, are implemented within the present invention.

Tokenization is implemented by a text analyzer which 50 treats the text as a series of tokens or useful meaningful units that are larger than individual characters, but smaller than phrases and sentences. These include words, separable parts of words, and punctuation. Each token is associated with an process of segmentation which extracts the individual tokens from the input text and keeps track of the offset where each token originated in the input text. The tokenizer output lists the offset and category for each token. In the next phase of the text analysis, the tagger uses a built-in morphological 60 analyzer to look up each word/token in a phrase or sentence and internally lists all parts of speech. The output is the input string with each token tagged with a parts of speech notation. Finally the grouper which functions as a phrase extractor or phrase analyzer, determines which groups of words 65 form phrases. These three operations which are the foundations for any modern linguistic processing schemes, are fully

18

implemented in optimized algorithms for determining the single-best possible answer to the user's question.

SQL Database and Full-Text Query

Another key component of present system is a SQLdatabase. This database is used to store text, specifically the answer-question pairs are stored in full-text tables of the database. Additionally, the full-text search capability of the database allows full-text searches to be carried out.

While a large portion of all digitally stored information is in the form of unstructured data, primarily text, it is now possible to store this textual data in traditional database systems in character-based columns such as varchar and text. In order to effectively retrieve textual data from the database, techniques have to be implemented to issue queries against textual data and to retrieve the answers in a meaningful way where it provides the answers as in the case of the NLQS system.

There are two major types of textual searches: Property— This search technology first applies filters to documents in order to extract properties such as author, subject, type, word count, printed page count, and time last written, and then issues searches against those properties; Full-text-this search technology first creates indexes of all non-noise words in the documents, and then uses these indexes to support linguistic searches and proximity searches.

Two additional technologies are also implemented in this particular RDBMs: SQL Server also have been integrated: A Search service—a full-text indexing and search service that is called both index engine and search, and a parser that accepts full-text SQL extensions and maps them into a form that can be processed by the search engine.

The four major aspects involved in implementing full-text retrieval of plain-text data from a full-text-capable database are: Managing the definition of the tables and columns that are registered for full-text searches; Indexing the data in registered columns—the indexing process scans the character streams, determines the word boundaries (this is called word breaking), removes all noise words (this also is called stop words), and then populates a full-text index with the remaining words; Issuing queries against registered columns for populated full-text indexes; Ensuring that subsequent changes to the data in registered columns gets propagated to the index engine to keep the full-text indexes synchronized.

The underlying design principle for the indexing, querying, and synchronizing processes is the presence of a fulltext unique key column (or single-column primary key) on all tables registered for full-text searches. The full-text index contains an entry for the non-noise words in each row together with the value of the key column for each row.

When processing a full-text search, the search engine returns to the database the key values of the rows that match the search criteria.

The full-text administration process starts by designating offset and a length. The first phase of tokenization is the 55 a table and its columns of interest for full-text search. Customized NLQS stored procedures are used first to register tables and columns as eligible for full-text search. After that, a separate request by means of a stored procedure is issued to populate the full-text indexes. The result is that the underlying index engine gets invoked and asynchronous index population begins. Full-text indexing tracks which significant words are used and where they are located. For example, a full-text index might indicate that the word "NLQS" is found at word number 423 and word number 982 in the Abstract column of the DevTools table for the row associated with a ProductID of 6. This index structure supports an efficient search for all items containing indexed

19

words as well as advanced search operations, such as phrase searches and proximity searches. (An example of a phrase search is looking for "white elephant," where "white" is followed by "elephant". An example of a proximity search is looking for "big" and "house" where "big" occurs near 5 "house".) To prevent the full-text index from becoming bloated, noise words such as "a," "and," and "the" are ignored.

Extensions to the Transact-SQL language are used to construct full-text queries. The two key predicates that are 10 used in the NLOS are CONTAINS and FREETEXT.

The CONTAINS predicate is used to determine whether or not values in full-text registered columns contain certain words and phrases. Specifically, this predicate is used to search for:

A word or phrase.

The prefix of a word or phrase.

A word or phrase that is near another.

A word that is an inflectional form of another (for example, "drive" is the inflectional stem of "drives," 20 "drove," "driving," and "driven").

A set of words or phrases, each of which is assigned a different weighting.

The relational engine within SQL Server recognizes the CONTAINS and FREETEXT predicates and performs some 25 minimal syntax and semantic checking, such as ensuring that the column referenced in the predicate has been registered for full-text searches. During query execution, a full-text predicate and other relevant information are passed to the full-text search component. After further syntax and 30 semantic validation, the search engine is invoked and returns the set of unique key values identifying those rows in the table that satisfy the full-text search condition. In addition to the FREETEXT and CONTAINS, other predicates such as AND, LIKE, NEAR are combined to create the customized 35 NLQS SQL construct.

Full-Text Query Architecture of the SQL Database

The full-text query architecture is comprised of the following several components—Full-Text Query component, the SQL Server Relational Engine, the Full-Text provider and the Search Engine.

The Full-Text Query component of the SQL database accept a full-text predicate or rowset-valued function from the SQL Server; transform parts of the predicate into an internal format, and sends it to Search Service, which returns the matches in a rowset. The rowset is then sent back to SQL Server. SQL Server uses this information to create the resultset that is then returned to the submitter of the query.

The SQL Server Relational Engine accepts the CONTAINS and FREETEXT predicates as well as the CONTAINSTABLE() and FREETEXTTABLE() rowset-valued functions. During parse time, this code checks for conditions such as attempting to query a column that has not been registered for full-text search. If valid, then at run time, the ft_search_condition and context information is sent to the full-text provider. Eventually, the full-text provider returns a rowset to SQL Server, which is used in any joins (specified or implied) in the original query. The Full-Text Provider parses and validates the ft_search_condition, constructs the appropriate internal representation of the full-text search condition, and then passes it to the search engine. The result is returned to the relational engine by means of a rowset of rows that satisfy ft_search_condition.

Client Side System 150

The architecture of client-side system 150 of Natural Language Query System 100 is illustrated in greater detail in

20

FIGS. 2A-2C. Referring to FIG. 2A, the three main processes effectuated by Client System 150 are illustrated as follows: Initialization process 200A consisting of SRE 201, Communication 202 and Microsoft (MS) Agent 203 routines; at FIG. 2B an iterative process 200B consisting of two sub-routines: a) Receive User Speech 208—made up of SRE 204 and Communication 205; and b) Receive Answer from Server 207—made up of MS Speak Agent 206, Communication 209, Voice data file 210 and Text to Speech Engine 211. Finally, in FIG. 2C un-initialization process 200C is made up of three sub-routines: SRE 212, Communication 213, and MS Agent 214. Each of the above three processes are described in detail in the following paragraphs. It will be appreciated by those skilled in the art that the particular 15 implementation for such processes and routines will vary from client platform to platform, so that in some environments such processes may be embodied in hard-coded routines executed by a dedicated DSP, while in others they may be embodied as software routines executed by a shared host processor, and in still others a combination of the two may be used.

Initialization at Client System 150

The initialization of the Client System **150** is illustrated in FIG. **2**D and is comprised generally of **3** separate initializing processes: client-side Speech Recognition Engine **220**A, MS Agent **220**B and Communication processes **220**C.

Initialization of Speech Recognition Engine 220A

Speech Recognition Engine 155 is initialized and configured using the routines shown in 220A. First, an SRE COM Library is initialized. Next, memory 220 is allocated to hold Source and Coder objects, are created by a routine 221. Loading of configuration file 221A from configuration data file 221B also takes place at the same time that the SRE Library is initialized. In configuration file 221B, the type of the input of Coder and the type of the output of the Coder are declared. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches. Accordingly, they are not discussed in detail herein. Next, Speech and Silence components of an utterance are calibrated using a routine 222, in a procedure that is also well-known in the art. To calibrate the speech and silence components, the user preferably articulates a sentence that is displayed in a text box on the screen. The SRE library then estimates the noise and other parameters required to find silence and speech elements of future user utterances.

Initialization of MS Agent 220B

The software code used to initialize and set up a MS Agent 220B is also illustrated in FIG. 2D. The MS Agent 220B routine is responsible for coordinating and handling the actions of the animated agent 157 (FIG. 1). This initialization thus consists of the following steps:

- Initialize COM library 223. This part of the code initializes the COM library, which is required to use ActiveX Controls, which controls are well-known in the art.
- Create instance of Agent Server 224—this part of the code creates an instance of Agent ActiveX control.
- 3. Loading of MS Agent 225—this part of the code loads MS Agent character from a specified file 225A containing general parameter data for the Agent Character, such as the overall appearance, shape, size, etc.
- 4. Get Character Interface 226—this part of the code gets an appropriate interface for the specified character; for

example, characters may have different control/interaction capabilities that can be presented to the user.

21

- 5. Add Commands to Agent Character Option 227—this part of the code adds commands to an Agent Properties sheet, which sheet can be accessed by clicking on the 5 icon that appears in the system tray, when the Agent character is loaded e.g., that the character can Speak, how he/she moves, TTS Properties, etc.
- 6. Show the Agent Character 228—this part of the code displays the Agent character on the screen so it can be 10 seen by the user;
- 7. AgentNotifySink—to handle events. This part of the code creates AgentNotifySink object 229, registers it at 230 and then gets the Agent Properties interface 231. The property sheet for the Agent character is assigned 15 using routine 232.
- 8. Do Character Animations 233—This part of the code plays specified character animations to welcome the user to NLOS 100.

The above then constitutes the entire sequence required to 20 initialize the MS Agent. As with the SRE routines, the MS Agent routines can be implemented in any suitable and conventional fashion by those skilled in the art based on the present teachings. The particular structure, operation, etc. of such routines is not critical, and thus they are not discussed 25 in detail herein.

In a preferred embodiment, the MS Agent is configured to have an appearance and capabilities that are appropriate for the particular application. For instance, in a remote learning application, the agent has the visual form and mannerisms/ attitude/gestures of a college professor. Other visual props (blackboard, textbook, etc.) may be used by the agent and presented to the user to bring to mind the experience of being in an actual educational environment. The characteristics of the agent may be configured at the client side 150, 35 and/or as part of code executed by a browser program (not shown) in response to configuration data and commands from a particular web page. For example, a particular website offering medical services may prefer to use a visual image of a doctor. These and many other variations will be 40 apparent to those skilled in the art for enhancing the humanlike, real-time dialog experience for users.

Initialization of Communication Link 160A

The initialization of Communication Link 160A is shown 45 with reference to process 220°C FIG. 2D. Referring to FIG. 2D, this initialization consists of the following code components: Open INTERNET Connection 234—this part of the code opens an INTERNET Connection and sets the parameter for the connection. Then Set Callback Status routine 235 sets the callback status so as to inform the user of the status of connection. Finally Start New HTTP INTERNET Session 236 starts a new INTERNET session. The details of Communications Link 160 and the set up process 220C are not critical, and will vary from platform to platform. Again, in 55 some cases, users may use a low-speed dial-up connection, a dedicated high speed switched connection (T1 for example), an always-on xDSL connection, a wireless connection, and the like.

Iterative Processing of Queries/Answers

As illustrated in FIG. 3, once initialization is complete, an iterative query/answer process is launched when the user presses the Start Button to initiate a query. Referring to FIG. 3, the iterative query/answer process consists of two main 65 sub-processes implemented as routines on the client side system 150: Receive User Speech 240 and Receive User

22

Answer 243. The Receive User Speech 240 routine receives speech from the user (or another audio input source), while the Receive User Answer 243 routine receives an answer to the user's question in the form of text from the server so that it can be converted to speech for the user by text-to-speech engine 159. As used herein, the term "query" is referred to in the broadest sense to refer, to either a question, a command, or some form of input used as a control variable by the system. For example, a query may consist of a question directed to a particular topic, such as "what is a network" in the context of a remote learning application. In an e-commerce application a query might consist of a command to "list all books by Mark Twain" for example. Similarly, while the answer in a remote learning application consists of text that is rendered into audible form by the text to speech engine 159, it could also be returned as another form of multi-media information, such as a graphic image, a sound file, a video file, etc. depending on the requirements of the particular application. Again, given the present teachings concerning the necessary structure, operation, functions, performance, etc., of the client-side Receive User Speech 240 and Receiver User Answer 243 routines, one of ordinary skill in the art could implement such in a variety of ways.

Receive User Speech-As illustrated in FIG. 3, the Receive User Speech routine 240 consists of a SRE 241 and a Communication 242 process, both implemented again as routines on the client side system 150 for receiving and partially processing the user's utterance. SRE routine 241 uses a coder 248 which is prepared so that a coder object receives speech data from a source object. Next the Start Source 249 routine is initiated. This part of the code initiates data retrieval using the source Object which will in turn be given to the Coder object. Next, MFCC vectors 250 are extracted from the Speech utterance continuously until silence is detected. As alluded to earlier, this represents the first phase of processing of the input speech signal, and in a preferred embodiment, it is intentionally restricted to merely computing the MFCC vectors for the reasons already expressed above. These vectors include the 12 cepstral coefficients and the RMS energy term, for a total of 13 separate numerical values for the partially processed speech

In some environments, nonetheless, it is conceivable that the MFCC delta parameters and MFCC acceleration parameters can also be computed at client side system 150, depending on the computation resources available, the transmission bandwidth in data link 160A available to server side system 180, the speed of a transceiver used for carrying data in the data link, etc. These parameters can be determined automatically by client side system upon initializing SRE 155 (using some type of calibration routine to measure resources), or by direct user control, so that the partitioning of signal processing responsibilities can be optimized on a case-by-case basis. In some applications, too, server side system 180 may lack the appropriate resources or routines for completing the processing of the speech input signal. Therefore, for some applications, the allocation of signal processing responsibilities may be partitioned differently, to the point where in fact both phases of the speech signal processing may take place at client side system 150 so that the speech signal is completely—rather than partiallyprocessed and transmitted for conversion into a query at server side system 180.

Again in a preferred embodiment, to ensure reasonable accuracy and real-time performance from a query/response perspective, sufficient resources are made available in a client side system so that 100 frames per second of speech

23

data can be partially processed and transmitted through link 160A. Since the least amount of information that is necessary to complete the speech recognition process (only 13 coefficients) is sent, the system achieves a real-time performance that is believed to be highly optimized, because other 5 latencies (i.e., client-side computational latencies, packet formation latencies, transmission latencies) are minimized. It will be apparent that the principles of the present invention can be extended to other SR applications where some other methodology is used for breaking down the speech input signal by an SRE (i.e., non-MFCC based). The only criteria is that the SR processing be similarly dividable into multiple phases, and with the responsibility for different phases being handled on opposite sides of link 160A depending on overall system performance goals, requirements and the like. This 15 functionality of the present invention can thus be achieved on a system-by-system basis, with an expected and typical amount of optimization being necessary for each particular implementation.

Thus, the present invention achieves a response rate 20 performance that is tailored in accordance with the amount of information that is computed, coded and transmitted by the client side system **150**. So in applications where real-time performance is most critical, the least possible amount of extracted speech data is transmitted to reduce these 25 latencies, and, in other applications, the amount of extracted speech data that is processed, coded and transmitted can be varied.

Communication—transmit communication module **242** is used to implement the transport of data from the client to the ³⁰ server over the data link **160**A, which in a preferred embodiment is the INTERNET. As explained above, the data consists of encoded MFCC vectors that will be used at then server-side of the Speech Recognition engine to complete the speech recognition decoding. The sequence of the communication is as follows:

- OpenHTTPRequest **251**—this part of the code first converts MFCC vectors to a stream of bytes, and then processes the bytes so that it is compatible with a protocol known as HTTP. This protocol is well-known in the art, and it is apparent that for other data links another suitable protocol would be used.
- Encode MFCC Byte Stream 251—this part of the code encodes the MFCC vectors, so that they can be sent to the server via HTTP.
- Send data 252—this part of the code sends MFCC vectors to the server using the INTERNET connection and the HTTP protocol.

Wait for the Server Response 253—this part of the code monitors the data link 160A a response from server side system 180 arrives. In summary, the MFCC parameters are extracted or observed on-the-fly from the input speech signal. They are then encoded to a HTTP byte stream and sent in a streaming fashion to the server before the silence is detected—i.e. sent to server side system 180 before the utterance is complete. This aspect of the invention also facilitates a real-time behavior, since data can be transmitted and processed even while the user is still speaking.

Receive Answer from Server 243 is comprised of the 60 following modules as shown in FIG. 3.: MS Agent 244, Text-to-Speech Engine 245 and receive communication modules 246. All three modules interact to receive the answer from server side system 180. As illustrated in FIG. 3, the receive communication process consists of three 65 separate processes implemented as a receive routine on client side system 150: a Receive the Best Answer 258

24

receives the best answer over data link 160B (the HTTP communication channel). The answer is de-compressed at 259 and then the answer is passed by code 260 to the MS Agent 244, where it is received by code portion 254. A routine 255 then articulates the answer using text-to-speech engine 257. Of course, the text can also be displayed for additional feedback purposes on a monitor used with client side system 150. The text to speech engine uses a natural language voice data As file 256 associated with it that is appropriate for the particular language application (i.e., English, French, German, Japanese, etc.). As explained earlier when the answer is something more than text, it can be treated as desired to provide responsive information to the user, such as with a graphics image, a sound, a video clip,

Uninitialization

The un-initialization routines and processes are illustrated in FIG. 4. Three functional modules are used for uninitializing the primary components of the client side system 150; these include SRE 270, Communications 271 and MS Agent 272 un-initializing routines. To un-initialize SRE 220A, memory that was allocated in the initialization phase is de-allocated by code 273 and objects created during such initialization phase are deleted by code 274. Similarly, as illustrated in FIG. 4, to un-initialize Communications module 220C the INTERNET connection previously established with the server is closed by code portion 275 of the Communication Un-initialization routine 271. Next the INTER-NET session created at the time of initialization is also closed by routine 276. For the un-initialization of the MS Agent 220B, as illustrated in FIG. 4, MS Agent Un-initialization routine 272 first releases the Commands Interface 227 using routine 277. This releases the commands added to the property sheet during loading of the agent character by routine 225. Next the Character Interface initialized by routine 226 is released by routine 278 and the Agent is unloaded at 279. The Sink Object Interface is then also released 280 followed by the release of the Property Sheet Interface 281. The Agent Notify Sink 282 then un-registers the Agent and finally the Agent Interface 283 is released which releases all the resources allocated during initialization steps identified in FIG. 2D.

It will be appreciated by those skilled in the art that the particular implementation for such un-initialization processes and routines in FIG. 4 will vary from client platform to client platform, as for the other routines discussed above. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches without undue effort. Accordingly, they are not discussed in detail herein.

Description of Server Side System 180

Introduction

A high level flow diagram of the set of preferred processes implemented on server side system 180 of Natural Language Query System 100 is illustrated in FIG. 11A through FIG. 11C. In a preferred embodiment, this process consists of a two step algorithm for completing the processing of the speech input signal, recognizing the meaning of the user's query, and retrieving an appropriate answer/response for such query.

The 1st step as illustrated in FIG. 11A can be considered a high-speed first-cut pruning mechanism, and includes the following operations: after completing processing of the speech input signal, the user's query is recognized at step 1101, so that the text of the query is simultaneously sent to

25

Natural Language Engine 190 (FIG. 1) at step 1107, and to DB Engine 186 (also FIG. 1) at step 1102. By "recognized" in this context it is meant that the user's query is converted into a text string of distinct native language words through the HMM technique discussed earlier.

At NLE 190, the text string undergoes morphological linguistic processing at step 1108: the string is tokenized the tags are tagged and the tagged tokens are grouped Next the noun phrases (NP) of the string are stored at 1109, and also copied and transferred for use by DB Engine 186 during a DB Process at step 1110. As illustrated in FIG. 11A, the string corresponding to the user's query which was sent to the DB Engine 186 at 1102, is used together with the NP received from NLE 190 to construct an SQL Query at step 1103. Next, the SQL query is executed at step 1104, and a record set of potential questions corresponding to the user's query are received as a result of a full-text search at 1105, which are then sent back to NLE 190 in the form of an array at step 1106.

As can be seen from the above, this first step on the server side processing acts as an efficient and fast pruning mechanism so that the universe of potential "hits" corresponding to the user's actual query is narrowed down very quickly to a manageable set of likely candidates in a very short period of 25 time.

Referring to FIG. 11B, in contrast to the first step above, the 2^{nd} step can be considered as the more precise selection portion of the recognition process. It begins with linguistic processing of each of the stored questions in the array 30 returned by the full-text search process as possible candidates representing the user's query. Processing of these stored questions continues in NLE 190 as follows: each question in the array of questions corresponding to the record set returned by the SQL full-text search undergoes 35 morphological linguistic processing at step 1111: in this operation, a text string corresponding to the retrieved candidate question is tokenized, the tags are tagged and the tagged tokens are grouped. Next, noun phrases of the string are computed and stored at step 1112. This process continues 40 iteratively at point 1113, and the sequence of steps at 1118, 1111, 1112, 1113 are repeated so that an NP for each retrieved candidate question is computed and stored. Once an NP is computed for each of the retrieved candidate questions of the array, a comparison is made between each 45 such retrieved candidate question and the user's query based on the magnitude of the NP value at step 1114. This process is also iterative in that steps 1114, 1115, 1116, 1119 are repeated so that the comparison of the NP for each retrieved candidate question with that of the NP of the user's query is $_{50}$ completed. When there are no more stored questions in the array to be processed at step 1117, the stored question that has the maximum NP relative to the user's query, is identified at 1117A as the stored question which best matches the

Notably, it can be seen that the second step of the recognition process is much more computationally intensive than the first step above, because several text strings are tokenized, and a comparison is made of several NPs. This that the first step has already quickly and efficiently reduced the candidates to be evaluated to a significant degree. Thus, this more computationally intensive aspect of the present invention is extremely valuable, however because it yields extremely high accuracy in the overall query recognition 65 process. In this regard, therefore, this second step of the query recognition helps to ensure the overall accuracy of the

26

system, while the first step helps to maintain a satisfactory speed that provides a real-time feel for the user.

As illustrated in FIG. 11C, the last part of the query/ response process occurs by providing an appropriate matching answer/response to the user. Thus, an identity of a matching stored question is completed at step 1120. Next a file path corresponding to an answer of the identified matching question is extracted at step 1121. Processing continues so that the answer is extracted from the file path at 1122 and finally the answer is compressed and sent to client side system 150 at step 1123.

The discussion above is intended to convey a general overview of the primary components, operations, functions and characteristics of those portions of NLQS system 100 that reside on server side system 180. The discussion that follows describes in more detail the respective sub-systems.

Software Modules used in Server Side System 180

The key software modules used on server-side system 180 of the NLQS system are illustrated in FIG. 5. These include generally the following components: a Communication module 500-identified as CommunicationServer ISAPI 500A (which is executed by SRE Server-side 182—FIG. 1 and is explained in more detail below), and a database process DBProcess module 501 (executed by DB Engine 186—FIG. 1). Natural language engine module 500C (executed by NLE 190-FIG. 1) and an interface 500B between the NLE process module 500C and the DBProcess module 500B. As shown here, CommunicationServerISAPI 500A includes a server-side speech recognition engine and appropriate communication interfaces required between client side system 150 and server side system 180. As further illustrated in FIG. 5, server-side logic of Natural Language Query System 100 also can be characterized as including two dynamic link library components: CommunicationServerISAPI 500 and DBProcess 501. The CommunicationServerIASPI 500 is comprised of 3 sub-modules: Server-side Speech Recognition Engine module 500A; Interface module 500B between Natural Language Engine modules 500C and DBProcess 501; and the Natural Language Engine modules **500**C.

DB Process 501 is a module whose primary function is to connect to a SQL database and to execute an SQL query that is composed in response to the user's query. In addition, this module interfaces with logic that fetches the correct answer from a file path once this answer is passed to it from the Natural Language Engine module 500C.

Speech Recognition Sub-System 182 on Server-Side System 180

The server side speech recognition engine module 500A is a set of distributed components that perform the necessary functions and operations of speech recognition engine 182 (FIG. 1) at server-side 180. These components can be implemented as software routines that are executed by 55 server side 180 in conventional fashion. Referring to FIG. 4A, a more detailed break out of the operation of the speech recognition components 600 at the server-side can be seen as

Within a portion 601 of the server side SRE module 500A, would not be practical, nonetheless, if it were not for the fact 60 the binary MFCC vector byte stream corresponding to the speech signal's acoustic features extracted at client side system 150 and sent over the communication channel 160 is received. The MFCC acoustic vectors are decoded from the encoded HTTP byte stream as follows: Since the MFCC vectors contain embedded NULL characters, they cannot be transferred in this form to server side system 180 as such using HTTP protocol. Thus the MFCC vectors are first

27

encoded at client-side **150** before transmission in such a way that all the speech data is converted into a stream of bytes without embedded NULL characters in the data. At the very end of the byte stream a single NULL character is introduced to indicate the termination of the stream of bytes to be 5 transferred to the server over the INTERNET **160**A using HTTP protocol.

As explained earlier, to conserve latency time between the client and server, a smaller number of bytes (just the 13 MFCC coefficients) are sent from client side system 150 to 10 server side system 180. This is done automatically for each platform to ensure uniformity, or can be tailored by the particular application environment—i.e., such as where it is determined that it will take less time to compute the delta and acceleration coefficients at the server (26 more calcu- 15 lations), than it would take to encode them at the client, transmit them, and then decode them from the HTTP stream. In general, since server side system 180 is usually better equipped to calculate the MFCC delta and acceleration parameters, this is a preferable choice. Furthermore, there is 20 generally more control over server resources compared to the client's resources, which means that future upgrades, optimizations, etc., can be disseminated and shared by all to make overall system performance more reliable and predictable. So, the present invention can accommodate even the 25 worst-case scenario where the client's machine may be quite thin and may just have enough resources to capture the speech input data and do minimal processing.

Dictionary Preparation & Grammar Files

Referring to FIG. 4A, within code block 605, various options selected by the user (or gleaned from the user's status within a particular application) are received. For instance, in the case of a preferred remote learning system, Course, Chapter and/or Section data items are communi- 35 cated. In the case of other applications (such as e-commerce) other data options are communicated, such as the Product Class, Product Category, Product Brand, etc. loaded for viewing within his/her browser. These selected options are based on the context experienced by the user during an 40 interactive process, and thus help to limit and define the scope—i.e. grammars and dictionaries that will be dynamically loaded to speech recognition engine 182 (FIG. 1) for Viterbi decoding during processing of the user speech utterance. For speech recognition to be optimized both grammar 45 and dictionary files are used in a preferred embodiment. A Grammar file supplies the universe of available user queries; i.e., all the possible words that are to be recognized. The Dictionary file provides phonemes (the information of how a word is pronounced—this depends on the specific native 50 language files that are installed—for example, UK English or US English) of each word contained in the grammar file. It is apparent that if all the sentences for a given environment that can be recognized were contained in a single grammar file then recognition accuracy would be deteriorated and the 55 loading time alone for such grammar and dictionary files would impair the speed of the speech recognition process.

To avoid these problems, specific grammars are dynamically loaded or actively configured as the current grammar according to the user's context, i.e., as in the case of a remote 60 learning system, the Course, Chapter and/or Section selected. Thus the grammar and dictionary files are loaded dynamically according to the given Course, Chapter and/or Section as dictated by the user, or as determined automatically by an application program executed by the user.

The second code block 602 implements the initialization of Speech Recognition engine 182 (FIG. 1). The MFCC

28

vectors received from client side system 150 along with the grammar filename and the dictionary file names are introduced to this block to initialize the speech decoder.

As illustrated in FIG. 4A, the initialization process 602 uses the following sub-routines: A routine 602a for loading an SRE library. This then allows the creation of an object identified as External Source with code 602b using the received MFCC vectors. Code 602c allocates memory to hold the recognition objects. Routine 602d then also creates and initializes objects that are required for the recognition such as: Source, Coder, Recognizer and Results Loading of the Dictionary created by code 602e, Hidden Markov Models (HMMs) generated with code 602f; and Loading of the Grammar file generated by routine 602g.

Speech Recognition 603 is the next routine invoked as illustrated in FIG. 4A, and is generally responsible for completing the processing of the user speech signals input on the client side 150, which, as mentioned above, are preferably only partially processed (i.e., only MFCC vectors are computed during the first phase) when they are transmitted across link 160. Using the functions created in External Source by subroutine 602b, this code reads MFCC vectors, one at a time from an External Source 603a, and processes them in block 603b to realize the words in the speech pattern that are symbolized by the MFCC vectors captured at the client. During this second phase, an additional 13 delta coefficients and an additional 13 acceleration coefficients are computed as part of the recognition process to obtain a total of 39 observation vectors O, referred to earlier. Then, using a set of previously defined Hidden Markov Models (HMMs), the words corresponding to the user's speech utterance are determined in the manner described earlier. This completes the word "recognition" aspect of the query processing, which results are used further below to complete the query processing operations.

It will be appreciated by those skilled in the art that the distributed nature and rapid performance of the word recognition process, by itself, is extremely useful and may be implemented in connection with other environments that do not implicate or require additional query processing operations. For example, some applications may simply use individual recognized words for filling in data items on a computer generated form, and the aforementioned systems and processes can provide a rapid, reliable mechanism for doing so.

Once the user's speech is recognized, the flow of SRE 182 passes to Un-initialize SRE routine 604 where the speech engine is un-initialized as illustrated. In this block all the objects created in the initialization block are deleted by routine 604a, and memory allocated in the initialization block during the initialization phase are removed by routine 604b.

Again, it should be emphasized that the above are merely illustrative of embodiments for implementing the particular routines used on a server side speech recognition system of the present invention. Other variations of the same that achieve the desired functionality and objectives of the present invention will be apparent from the present teachings.

Database Processor 186 Operation—DBProcess

Construction of an SQL Query used as part of the user query processing is illustrated in FIG. 4B, a SELECT SQL statement is preferably constructed using a conventional CONTAINS predicate. Module 950 constructs the SQL query based on this SELECT SQL statement, which query is used for retrieving the best suitable question stored in the

29

database corresponding to the user's articulated query, (designated as Question here). A routine 951 then concatenates a table name with the constructed SELECT statement. Next, the number of words present in each Noun Phrase of Question asked by the user is calculated by routine 952. 5 Then memory is allocated by routine 953 as needed to accommodate all the words present in the NP. Next a word List (identifying all the distinct words present in the NP) is obtained by routine 954. After this, this set of distinct words are concatenated by routine 955 to the SQL Query separated with a NEAR () keyword. Next, the AND keyword is concatenated to the SQL Query by routine 956 after each NP. Finally memory resources are freed by code 957 so as to allocate memory to store the words received from NP for any next iteration. Thus, at the end of this process, a 15 completed SQL Query corresponding to the user's articulated question is generated.

Connection to SQL Server—As illustrated in FIG. 4C, after the SQL Query is constructed by routine 710, a routine 711 implements a connection to the query database 717 to 20 continue processing of the user query. The connection sequence and the subsequent retrieved record set is implemented using routines 700 which include the following:

- 1. Server and database names are assigned by routine **711A** to a DBProcess member variable
- 2. A connection string is established by routine 711B;
- The SQL Server database is connected under control of code 711C
- 4. The SQL Query is received by routine 712A
- 5. The SQL Query is executed by code 712B
- Extract the total number of records retrieved by the query—713
- Allocate the memory to store the total number of paired questions—713
- Store the entire number of paired questions into an ³⁵ array—713

Once the Best Answer ID is received at 716 FIG. 4C, from the NLE 14 (FIG. 5), the code corresponding 716C receives it passes it to code in 716B where the path of the Answer file is determined using the record number. Then the file is opened 716C using the path passed to it and the contents of the file corresponding to the answer is read. Then the answer is compressed by code in 716D and prepared for transmission over the communication channel 160B (FIG. 1).

NLOS Database 188—Table Organization

FIG. 6 illustrates a preferred embodiment of a logical structure of tables used in a typical NLQS database 188 (FIG. 1). When NLQS database 188 is used as part of NLQS query system 100 implemented as a remote learning/training 50 environment, this database will include an organizational multi-level hierarchy that consists typically of a Course 701, which is made of several chapters 702, 703, 704. Each of these chapters can have one or more Sections 705, 706, 707 as shown for Chapter 1. A similar structure can exist for 55 Chapter 2, Chapter 3 . . . Chapter N. Each section has a set of one or more question—answer pairs 708 stored in tables described in more detail below. While this is an appropriate and preferable arrangement for a training/learning application, it is apparent that other implementations would be 60 possible and perhaps more suitable for other applications such as e-commerce, e-support, INTERNET browsing, etc., depending on overall system parameters.

It can be seen that the NLQS database 188 organization is intricately linked to the switched grammar architecture 65 described earlier. In other words, the context (or environment) experienced by the user can be determined at any

30

moment in time based at the selection made at the section level, so that only a limited subset of question-answer pairs 708 for example are appropriate for section 705. This in turn means that only a particular appropriate grammar for such question-answer pairs may be switched in for handling user queries while the user is experiencing such context. In a similar fashion, an e-commerce application for an INTER-NET based business may consist of a hierarchy that includes a first level "home" page 701 identifying user selectable options (product types, services, contact information, etc.), a second level may include one or more "product types" pages 702, 703, 704, a third page may include particular product models 705, 706, 707, etc., and with appropriate question-answer pairs 708 and grammars customized for handling queries for such product models. Again, the particular implementation will vary from application to application, depending on the needs and desires of such business, and a typical amount of routine optimization will be necessary for each such application.

Table Organization

In a preferred embodiment, an independent database is used for each Course. Each database in turn can include three types of tables as follows: a Master Table as illustrated in FIG. 7A, at least one Chapter Table as illustrated in FIG. 7B and at least one Section Table as illustrated in FIG. 7C.

As illustrated in FIG. 7A, a preferred embodiment of a Master Table has six columns—Field Name 701A, Data Type 702A, Size 703A, Null 704A, Primary Key 705A and Indexed 706A. These parameters are well-known in the art of database design and structure. The Master Table has only two fields—Chapter Name 707A and Section Name 708A. Both ChapterName and Section Name are commonly indexed.

A preferred embodiment of a Chapter Table is illustrated in FIG. 7B. As with the Master Table, the Chapter Table has six (6) columns—Field Name 720, Data Type 721, Size 722, Null 723, Primary Key 724 and Indexed 725. There are nine (9) rows of data however, in this case,—Chapter_ID 726, Answer_ID 727, Section Name 728, Answer_Title 729, PairedQuestion 730, AnswerPath 731, Creator 732, Date of Creation 733 and Date of Modification 734.

An explanation of the Chapter Table fields is provided in FIG. 7C. Each of the eight (8) Fields 720 has a description 45 735 and stores data corresponding to:

- AnswerID 727—an integer that is automatically incremented for each answer given for user convenience
- Section_Name 728—the name of the section to which the particular record belongs. This field along with the AnswerID is used as the primary key
- Answer_Title **729**—A short description of the title of the answer to the user query
- PairedQuestion 730—Contains one or more combinations of questions for the related answers whose path is stored in the next column AnswerPath
- AnswerPath 731—contains the path of a file, which contains the answer to the related questions stored in the previous column; in the case of a pure question/answer application, this file is a text file, but, as mentioned above, could be a multi-media file of any kind transportable over the data link 160

Creator 732—Name of Content Creator

- Date_of_Creation 733—Date on which content was created
- Date of Modification **734**—Date on which content was changed or modified

31

A preferred embodiment of a Section Table is illustrated in FIG. 7D. The Section Table has six (6) columns—Field Name 740, Data Type 741, Size 742, Null 743, Primary Key 744 and Indexed 745. There are seven (7) rows of data— Answer_ID 746, Answer_Title 747, PairedQuestion 748, 5 AnswerPath 749, Creator 750, Date of Creation 751 and Date of Modification 752. These names correspond to the same fields, columns already described above for the Master Table and Chapter Table.

Again, this is a preferred approach for the specific type of 10 learning/training application described herein. Since the number of potential applications for the present invention is quite large, and each application can be customized, it is expected that other applications (including other learning/ training applications) will require and/or be better accom- 15 modated by another table, column, and field structure/

Search Service and Search Engine—A query text search service is performed by an SQL Search System 1000 shown in FIG. 10. This system provides querying support to process 20 full-text searches. This is where full-text indexes reside.

In general, SQL Search System determines which entries in a database index meet selection criteria specified by a particular text query that is constructed in accordance with an articulated user speech utterance. The Index Engine 25 1011B is the entity that populates the Full-Text Index tables with indexes which correspond to the indexable units of text for the stored questions and corresponding answers. It scans through character strings, determines word boundaries, removes all noise words and then populates the full-text 30 index with the remaining words. For each entry in the full text database that meets the selection criteria, a unique key column value and a ranking value are returned as well. Catalog set 1013 is a file-system directory that is accessible only by an Administrator and Search Service 1010. Full-text 35 indexes 1014 are organized into full-text catalogs, which are referenced by easy to handle names. Typically, full-text index data for an entire database is placed into a single full-text catalog.

The schema for the full-text database as described (FIG. 40 7, FIG. 7A, FIG. 7B, FIG. 7C, FIG. 7D) is stored in the tables 1006 shown in FIG. 10. Take for example, the tables required to describe the structure the stored question/answer pairs required for a particular course. For each table-Course Table, Chapter Table, Section Table, there are 45 fields—column information that define each parameters that make up the logical structure of the table. This information is stored in User and System tables 1006. The key values corresponding to those tables are stored as Full-Text catalogs 1013. So when processing a full-text search, the search engine returns to the SQL Server the key values of the rows that match the search criteria. The relational engine then uses this information to respond to the query.

As illustrated in FIG. 10, a Full-Text Query Process is implemented as follows:

- 1. A query 1001 that uses a SQL full-text construct generated by DB processor 186 is submitted to SQL Relational Engine 1002.
- 2. Queries containing either a CONTAINS or FREETEXT predicate are rewritten by routine 1003 so that a respon- 60 sive rowset returned later from Full-Text Provider 1007 will be automatically joined to the table that the predicate is acting upon. This rewrite is a mechanism used to ensure that these predicates are a seamless extension to a traditional SQL Server. After the compiled query is 65 internally rewritten and checked for correctness in item 1003, the query is passed to RUN TIME module 1004.

32

- The function of module 1004 is to convert the rewritten SQL construct to a validated run-time process before it is sent to the Full-Text Provider, 1007.
- 3. After this, Full-Text Provider 1007 is invoked, passing the following information for the query:
 - a. A ft search condition parameter (this is a logical flag indicating a full text search condition)
 - b. A name of a full-text catalog where a full-text index of a table resides
 - c. A locale ID to be used for language (for example, word breaking)
 - d. Identities of a database, table, and column to be used in the query
 - e. If the query is comprised of more than one full-text construct; when this is the case Full-text provider 1007 is invoked separately for each construct.
- 4. SQL Relational Engine 1002 does not examine the contents of ft search condition. Instead, this information is passed along to Full-text provider 1007, which verifies the validity of the query and then creates an appropriate internal representation of the full-text search condition.
- 5. The query request/command 1008 is then passed to Querying Support 1011A.
- 6. Querying Support 1012 returns a rowset 1009 from Full-Text Catalog 1013 that contains unique key column values for any rows that match the full-text search criteria. A rank value also is returned for each row.
- 7. The rowset of key column values 1009 is passed to SQL Relational Engine 1002. If processing of the query implicates either a CONTAINSTABLE() or FREET-EXTTABLE() function, RANK values are returned; otherwise, any rank value is filtered out.
- 8. The rowset values 1009 are plugged into the initial query with values obtained from relational database 1006, and a result set 1015 is then returned for further processing to yield a response to the user.

At this stage of the query recognition process, the speech utterance by the user has already been rapidly converted into a carefully crafted text query, and this text query has been initially processed so that an initial matching set of results can be further evaluated for a final determination of the appropriate matching question/answer pair. The underlying principle that makes this possible is the presence of a full-text unique key column for each table that is registered for full-text searches. Thus when processing a full-text search, SQL Search Service 1010 returns to SQL server 1002 the key values of the rows that match the database. In maintaining these full-text databases 1013 and full text indexes 1014, the present invention has the unique characteristic that the full-text indices 1014 are not updated instantly when the full-text registered columns are updated. This operation is eliminated, again, to reduce recognition latency, increase response speed, etc. Thus, as compared to other database architectures, this updating of the full-text index tables, which would otherwise take a significant time, is instead done asynchronously at a more convenient time.

Interface between NLE 190 and DB Processor 188

The result set 1015 of candidate questions corresponding to the user query utterance are presented to NLE 190 for further processing as shown in FIG. 4D to determine a "best" matching question/answer pair. An NLE/DBProcessor interface module coordinates the handling of user queries, analysis of noun-phrases (NPs) of retrieved questions sets from the SQL query based on the user query, comparing the retrieved question NPs with the user query NP, etc.

33

between NLE 190 and DB Processor 188. So, this part of the server side code contains functions, which interface processes resident in both NLE block 190 and DB Processor block 188. The functions are illustrated in FIG. 4D; As seen here, code routine 880 implements functions to extract the 5 Noun Phrase (NP) list from the user's question. This part of the code interacts with NLE 190 and gets the list of Noun Phrases in a sentence articulated by the user. Similarly, Routine 813 retrieves an NP list from the list of corresponding candidate/paired questions 1015 and stores these ques- 10 tions into an (ranked by NP value) array. Thus, at this point, NP data has been generated for the user query, as well as for the candidate questions 1015. As an example of determining the noun phrases of a sentence such as: "What issues have guided the President in considering the impact of foreign 15 trade policy on American businesses?" NLE 190 would return the following as noun phrases: President, issues, impact of foreign trade policy, American businesses, impact, impact of foreign trade, foreign trade policy, trade, trade policy, policy, businesses. The methodology 20 used by NLE 190 will thus be apparent to those skilled in the art from this set of noun phrases and noun sub-phrases generated in response to the example query.

Next, a function identified as Get Best Answer ID **815** is implemented. This part of the code gets a best answer ID 25 corresponding to the user's query. To do this, routines **813**A, **813**B first find out the number of Noun phrases for each entry in the retrieved set **1015** that match with the Noun phrases in the user's query. Then routine **815**a selects a final result record from the candidate retrieved set **1015** that 30 contains the maximum number of matching Noun phrases.

Conventionally, nouns are commonly thought of as "naming" words, and specifically as the names of "people, places, or things". Nouns such as John, London, and computer certainly fit this description, but the types of words classified 35 by the present invention as nouns is much broader than this. Nouns can also denote abstract and intangible concepts such as birth, happiness, evolution, technology, management, imagination, revenge, politics, hope, cookery, sport, and literacy. Because of the enormous diversity of nouns compared to other parts of speech, the Applicant has found that it is much more relevant to consider the noun phrase as a key linguistic metric. So, the great variety of items classified as nouns by the present invention helps to discriminate and identify individual speech utterances much easier and faster 45 than prior techniques disclosed in the art.

Following this same thought, the present invention also adopts and implements another linguistic entity—the word phrase—to facilitate speech query recognition. The basic structure of a word phrase—whether it be a noun phrase, 50 verb phrase, adjective phrase—is three parts—[pre-Head string], [Head] and [post-Head string]. For example, in the minimal noun phrase—"the children," "children" is classified as the Head of the noun phrase. In summary, because of the diversity and frequency of noun phrases, the choice of 55 noun phrase as the metric by which stored answer is linguistically chosen, has a solid justification in applying this technique to the English natural language as well as other natural languages. So, in sum, the total noun phrases in a speech utterance taken together operate extremely well as 60 unique type of speech query fingerprint.

The ID corresponding to the best answer corresponding to the selected final result record question is then generated by routine **815** which then returns it to DB Process shown in FIG. **4**C. As seen there, a Best Answer ID I is received by 65 routine **716**A, and used by a routine **716**B to retrieve an answer file path. Routine **716**C then opens and reads the

34

answer file, and communicates the substance of the same to routine 716D. The latter then compresses the answer file data, and sends it over data link 160 to client side system 150 for processing as noted earlier (i.e., to be rendered into audible feedback, visual text/graphics, etc.). Again, in the context of a learning/instructional application, the answer file may consist solely of a single text phrase, but in other applications the substance and format will be tailored to a specific question in an appropriate fashion. For instance, an "answer" may consist of a list of multiple entries corresponding to a list of responsive category items (i.e., a list of books to a particular author) etc. Other variations will be apparent depending on the particular environment.

Natural Language Engine 190

Again referring to FIG. 4D, the general structure of NL engine 190 is depicted. This engine implements the word analysis or morphological analysis of words that make up the user's query, as well as phrase analysis of phrases extracted from the query.

As illustrated in FIG. 9, the functions used in a morphological analysis include tokenizers 802A, stemmers 804A and morphological analyzers 806A. The functions that comprise the phrase analysis include tokenizers, taggers and groupers, and their relationship is shown in FIG. 8.

Tokenizer 802A is a software module that functions to break up text of an input sentence 801A into a list of tokens 803A. In performing this function, tokenizer 802A goes through input text 801A and treats it as a series of tokens or useful meaningful units that are typically larger than individual characters, but smaller than phrases and sentences. These tokens 803A can include words, separable parts of word and punctuation. Each token 803A is given an offset and a length. The first phase of tokenization is segmentation, which extracts the individual tokens from the input text and keeps track of the offset where each token originated from in the input text. Next, categories are associated with each token, based on its shape. The process of tokenization is well-known in the art, so it can be performed by any convenient application suitable for the present invention.

Following tokenization, a stemmer process 804A is executed, which can include two separate forms—inflectional and derivational, for analyzing the tokens to determine their respective stems 805A. An inflectional stemmer recognizes affixes and returns the word which is the stem. A derivational stemmer on the other hand recognizes derivational affixes and returns the root word or words. While stemmer 804A associates an input word with its stem, it does not have parts of speech information. Analyzer 806B takes a word independent of context, and returns a set of possible parts of speech 806A.

As illustrated in FIG. 8, phrase analysis 800 is the next step that is performed after tokenization. A tokenizer 802 generates tokens from input text 801. Tokens 803 are assigned to parts of a speech tag by a tagger routine 804, and a grouper routine 806 recognizes groups of words as phrases of a certain syntactic type. These syntactic types include for example the noun phrases mentioned earlier, but could include other types if desired such as verb phrases and adjective phrases. Specifically, tagger 804 is a parts-ofspeech disambiguator, which analyzes words in context. It has a built-in morphological analyzer (not shown) that allows it to identify all possible parts of speech for each token. The output of tagger 804 is a string with each token tagged with a parts-of-speech label 805. The final step in the linguistic process 800 is the grouping of words to form phrases 807. This function is performed by the grouper 806,

35

and is very dependent, of course, on the performance and output of tagger component 804.

Accordingly, at the end of linguistic processing **800**, a list of noun phrases (NP) **807** is generated in accordance with the user's query utterance. This set of NPs generated by NLE 5 **190** helps significantly to refine the search for the best answer, so that a single-best answer can be later provided for the user's question.

The particular components of NLE **190** are shown in FIG. **4**D, and include several components. Each of these components implement the several different functions required in NLE **190** as now explained.

Initialize Grouper Resources Object and the Library 900—this routine initializes the structure variables required to create grouper resource object and library. Specifically, it 15 initializes a particular natural language used by NLE 190 to create a Noun Phrase, for example the English natural language is initialized for a system that serves the English language market. In turn, it also creates the objects (routines) required for Tokenizer, Tagger and Grouper (discussed 20 above) with routines 900A, 900B, 900C and 900D respectively, and initializes these objects with appropriate values. It also allocates memory to store all the recognized Noun Phrases for the retrieved question pairs.

Tokenizing of the words from the given text (from the 25 query or the paired questions) is performed with routine 909B—here all the words are tokenized with the help of a local dictionary used by NLE 190 resources. The resultant tokenized words are passed to a Tagger routine 909C. At routine 909C, tagging of all the tokens is done and the output 30 is passed to a Grouper routine 909D.

The Grouping of all tagged token to form NP list is implemented by routine 909D so that the Grouper groups all the tagged token words and outputs the Noun Phrases.

Un-initializing of the grouper resources object and freeing 35 of the resources, is performed by routines 909EA, 909EB and 909EC. These include Token Resources, Tagger Resources and Grouper Resources respectively. After initialization, the resources are freed. The memory that was used to store all Noun Phrases are also de-allocated.

Additional Embodiments

In an e-commerce embodiment of the present invention as illustrated in FIG. 13, a web page 1300 contains typical 45 visible links such as Books 1310, Music 1320 so that on clicking the appropriate link the customer is taken to those pages. The web page may be implemented using HTML, a Java applet, or similar coding techniques which interact with the user's browser. For example, if customer wants to buy an 50 album C by Artist Albert, he traverses several web pages as follows: he first clicks on Music (FIG. 13, 1360), which brings up page 1400 where he/she then clicks on Records (FIG. 14, 1450). Alternatively, he/she could select CDs 1460, Videos 1470, or other categories of books 1410, music 55 1420 or help 1430. As illustrated in FIG. 15, this brings up another web page 1500 with links for Records 1550, with sub-categories—Artist 1560, Song 1570, Title 1580, Genre **1590**. The customer must then click on Artist **1560** to select the artist of choice. This displays another web page 1600 as 60illustrated in FIG. 16. On this page the various artists 1650 are listed as illustrated—Albert 1650, Brooks 1660, Charlie 1670, Whyte 1690 are listed under the category Artists 1650. The customer must now click on Albert 1660 to view the albums available for Albert. When this is done, another web 65 page is displayed as shown in FIG. 17. Again this web page 1700 displays a similar look and feel, but with the albums

36

available 1760, 1770, 1780 listed under the heading Titles 1750. The customer can also read additional information 1790 for each album. This album information is similar to the liner notes of a shrink-wrapped album purchased at a retail store. One Album A is identified, the customer must click on the Album A 1760. This typically brings up another text box with the information about its availability, price, shipping and handling charges etc.

When web page 1300 is provided with functionality of a NLQS of the type described above, the web page interacts with the client side and server side speech recognition modules described above. In this case, the user initiates an inquiry by simply clicking on a button designated Contact Me for Help 1480 (this can be a link button on the screen, or a key on the keyboard for example) and is then told by character 1440 about how to elicit the information required. If the user wants Album A by artist Albert, the user could articulate "Is Album A by Brooks available?" in much the same way they would ask the question of a human clerk at a brick and mortar facility. Because of the rapid recognition performance of the present invention, the user's query would be answered in real-time by character 1440 speaking out the answer in the user's native language. If desired, a readable word balloon 1490 could also be displayed to see the character's answer and so that save/print options can also be implemented. Similar appropriate question/answer pairs for each page of the website can be constructed in accordance with the present teachings, so that the customer is provided with an environment that emulates a normal conversational human-like question and answer dialog for all aspects of the web site. Character 1440 can be adjusted and tailored according to the particular commercial application, or by the user's own preferences, etc. to have a particular voice style (man, woman, young, old, etc.) to enhance the customer's experience.

In a similar fashion, an articulated user query might be received as part of a conventional search engine query, to locate information of interest on the INTERNET in a similar manner as done with conventional text queries. If a reason-40 ably close question/answer pair is not available at the server side (for instance, if it does not reach a certain confidence level as an appropriate match to the user's question) the user could be presented with the option of increasing the scope so that the query would then be presented simultaneously to one or more different NLEs across a number of servers, to improve the likelihood of finding an appropriate matching question/answer pair. Furthermore, if desired, more than one "match" could be found, in the same fashion that conventional search engines can return a number of potential "hits" corresponding to the user's query. For some such queries, of course, it is likely that real-time performance will not be possible (because of the disseminated and distributed processing) but the advantage presented by extensive supplemental question/answer database systems may be desirable for some users.

It is apparent as well that the NLQS of the present invention is very natural and saves much time for the user and the e-commerce operator as well. In an e-support embodiment, the customer can retrieve information quickly and efficiently, and without need for a live customer agent. For example, at a consumer computer system vendor related support site, a simple diagnostic page might be presented for the user, along with a visible support character to assist him/her. The user could then select items from a "symptoms" page (i.e., a "monitor" problem, a "keyboard" problem, a "printer" problem, etc.) simply by articulating such symptoms in response to prompting from the support char-

37

acter. Thereafter, the system will direct the user on a real-time basis to more specific sub-menus, potential solutions, etc. for the particular recognized complaint. The use of a programmable character thus allows the web site to be scaled to accommodate a large number of hits or customers without any corresponding need to increase the number of human resources and its attendant training issues.

As an additional embodiment, the searching for information on a particular web site may be accelerated with the use of the NLQS of the present invention. Additionally, a significant benefit is that the information is provided in a user-friendly manner through the natural interface of speech. The majority of web sites presently employ lists of frequently asked questions which the user typically wades item by item in order to obtain an answer to a question or issue. For example, as displayed in FIG. 13, the customer clicks on Help 1330 to initiate the interface with a set of lists. Other options include computer related items at 1370 and frequently asked questions (FAQ) at 1380.

As illustrated in FIG. 18, a web site plan for typical web page is displayed. This illustrates the number of pages that have to be traversed in order to reach the list of Frequently-Asked Questions. Once at this page, the user has to scroll and manually identify the question that matches his/her 25 query. This process is typically a laborious task and may or may not yield the information that answers the user's query. The present art for displaying this information is illustrated in FIG. 18. This figure identifies how the information on a typical web site is organized: the Help link (FIG. 13, 1330) 30 typically shown on the home page of the web page is illustrated shown on FIG. 18 as 1800. Again referring to FIG. 18, each sub-category of information is listed on a separate page. For example, 1810 lists sub-topics such as 'First Time Visitors', 'Search Tips', 'Ordering', 'Shipping', 35 'Your Account' etc. Other pages deal with 'Account information' 1860, 'Rates and Policies' 1850 etc. Down another level, there are pages that deal exclusively with a sub-sub topics on a specific page such as 'First Time Visitors' 1960, 'Frequently Asked Questions' 1950, 'Safe Shopping Guar- 40 antee' 1940, etc. So if a customer has a query that is best answered by going to the Frequently Asked Questions link, he or she has to traverse three levels of busy and cluttered screen pages to get to the Frequently Asked Questions page 1950. Typically, there are many lists of questions 1980 that 45 have to be manually scrolled through. While scrolling visually, the customer then has to visually and mentally match his or her question with each listed question. If a possible match is sighted, then that question is clicked and the answer then appears in text form which then is read

In contrast, the process of obtaining an answer to a question using a web page enabled with the present NLQS can be achieved much less laboriously and efficiently. The user would articulate the word "Help" (FIG. 13, 1330). This would immediately cause a character (FIG. 13, 1340) to 55 appear with the friendly response "May I be of assistance. Please state your question?". Once the customer states the question, the character would then perform an animation or reply "Thank you, I will be back with the answer soon". After a short period time (preferably not exceeding 5-7 60 seconds) the character would then speak out the answer to the user's question. As illustrated in FIG. 18 the answer would be the answer 1990 returned to the user in the form of speech is the answer that is paired with the question 1950. For example, the answer 1990: "We accept Visa, MasterCard 65 and Discover credit cards", would be the response to the query 2000 "What forms of payments do you accept?"

38

Another embodiment of the invention is illustrated in FIG. 12. This web page illustrates a typical website that employs NLOS in a web-based learning environment. As illustrated in FIG. 12, the web page in browser 1200, is divided into two or more frames. A character 1210 in the likeness of an instructor is available on the screen and appears when the student initiates the query mode either by speaking the word "Help" into a microphone (FIG. 2B, 215) or by clicking on the link 'Click to Speak' (FIG. 12, 1280). Character 1210 would then prompt the student to select a course 1220 from the drop down list 1230. If the user selects the course 'CPlusPlus', the character would then confirm verbally that the course "CPlusPlus" was selected. The character would then direct the student to make the next selection from the drop-down list 1250 that contains the selections for the chapters 1240 from which questions are available. Again, after the student makes the selection, the character 1210 confirms the selection by speaking. Next character 1210 prompts the student to select 'Section' 1260 of the chapter from which questions are available from the drop down list 1270. Again, after the student makes the selection, character 1210 confirms the selection by articulating the 'Section' 1260 chosen. As a prompt to the student, a list of possible questions appear in the list box 1291. In addition, tips 1290 for using the system are displayed. Once the selections are all made, the student is prompted by the character to ask the question as follows: "Please ask your query now". The student then speaks his query and after a short period of time, the character responds with the answer preceded by the question as follows: "The answer to your question . . . is as follows: . . . ". This procedure allows the student to quickly retrieve answers to questions about any section of the course and replaces the tedium of consulting books, and references or indices. In short, it is can serve a number of uses from being a virtual teacher answering questions on-the-fly or a flash card substitute.

From preliminary data available to the inventors, it is estimate that the system can easily accommodate 100-250 question/answer pairs while still achieving a real-time feel and appearance to the user (i.e., less than 10 seconds of latency, not counting transmission) using the above described structures and methods. It is expected, of course, that these figures will improve as additional processing speed becomes available, and routine optimizations are employed to the various components noted for each particular environment.

Again, the above are merely illustrative of the many possible applications of the present invention, and it is 50 expected that many more web-based enterprises, as well as other consumer applications (such as intelligent, interactive toys) can utilize the present teachings. Although the present invention has been described in terms of a preferred embodiment, it will be apparent to those skilled in the art that many alterations and modifications may be made to such embodiments without departing from the teachings of the present invention. It will also be apparent to those skilled in the art that many aspects of the present discussion have been simplified to give appropriate weight and focus to the more germane aspects of the present invention. The microcode and software routines executed to effectuate the inventive methods may be embodied in various forms, including in a permanent magnetic media, a non-volatile ROM, a CD-ROM, or any other suitable machine-readable format. Accordingly, it is intended that the all such alterations and modifications be included within the scope and spirit of the invention as defined by the following claims.

What is claimed is:

1. A method of interacting with a user of a natural language query system comprising the steps of:

39

- a) providing a speech recognition engine adapted to recognize a first set of words and/or phrases from a user 5 during an interactive speech session;
 - wherein said first set of words and/or phrases can include a natural language query presented as continuous natural language spoken data;
- b) providing a database of query/answer pairs concerning 10 one or more topics which can be responded to by the natural language query system during said interactive speech based session with a user;
- c) providing a natural language routine adapted to process said first set of words and/or phrases and identify a 15 response to said natural language query based on said query/answer pairs;
 - wherein said natural language routine is adapted to consider only a subset of said first set of words and/or phrases, and further can consider words and/ 20 or phrases in said natural language query which are not present in said query/answer pairs to determine said response;
- d) providing an interactive electronic agent coupled to said natural language routine and configured to:
 - provide a prompt to the user during said interactive speech based session with suggestions on queries which can be made to the natural language query system:
 - ii. provide a confirmation of a substance of said natural 30 language query;
 - iii. provide said response to the user from the natural language query routine.
- 2. The method of claim 1, wherein the interactive electronic agent is further configured to provide feedback while 35 said natural language routine is processing said user natural language query.
- 3. The method of claim 2, wherein at least one of said prompt, confirmation or response is processed by a text to speech engine and rendered into audible form for the user by 40 the interactive electronic agent.
- **4**. The method of claim **2**, wherein the interactive electronic agent automatically communicates in a voice with speech characteristics based on a native language used by the user.
- **5**. The method of claim **1**, wherein the interactive electronic agent is incorporated as part of an e-commerce Web page and presented within a Web browser program operating on a client device operated by the user.
- **6**. The method of claim **5** wherein the electronic interactive agent form is an animated character on a screen of the client device.
 - 7. The method of claim 1 further including a step: configuring perception related parameters of the electronic interactive agent based on preferences of said 55 user, including one of a gender, a visual appearance, and/or voice characteristics including one of pitch, volume and/or speed.
- **8**. The method of claim **1** wherein said subset of words and/or phrases in said natural language query can be 60 assigned different weightings determined by said natural language routine.
- **9.** The method of claim **1**, wherein said speech recognition engine is distributed between a client device and a server system which receives streaming speech data having 65 reduced latency data content before silence is detected and the utterance is complete.

40

- 10. The method of claim 1, wherein said response is selected based on comparing which of said subset of words and/or phrases in said speech-based query also appear in said query/answer pairs.
- 11. The method of claim 10, wherein a single response is selected based on determining which of said query/answer pairs has an optimal number of words and/or phrases overlapping with said natural language query.
- 12. The method of claim 1 further including providing a second natural language routine at a separate computing system adapted to assist with a response to said natural language query, such that said natural language query is recognized by said speech recognition engine, and multiple databases at different server systems can be considered in responding to natural language queries for a set of topic entries.
- 13. The method of claim 1 wherein the interactive electronic agent provides responses adjusted for a context experienced by the user.
- 14. The system of claim 1 wherein the interactive electronic agent provides responses adjusted for a context experienced by the user.
- 15. An electronic agent based natural language query system comprising:
 - a) a database of query/answer pairs concerning one or more topics which can be responded to by the natural language query system during an interactive speech based session with a user;
 - b) a speech recognition engine executing at a computing system and adapted to recognize a first set of words and/or phrases from a user during said interactive speech session;
 - wherein said first set of words and/or phrases can include a natural language query presented as continuous natural language spoken data;
 - c) a natural language routine executing at said computing system and adapted to process said words and/or phrases and identify a response to said natural language query derived from said query/answer pairs;
 - wherein said natural language routine is adapted to consider only a subset of said words and/or phrases, and can consider words and/or phrases in said natural language query which are not present in said query/ answer pairs to determine said response;
 - d) an interactive agent routine coupled to said natural language routine and for providing an interactive electronic agent configured and controlled by one or more additional second routines to:
 - provide a prompt to the user during said interactive speech based session with suggestions on queries which can be made to the natural language query system;
 - ii. provide a confirmation of a substance of said natural language query;
 - iii. provide said response to the user from the natural language routine.
- **16**. The system of claim **15**, wherein the interactive electronic agent is further configured to provide feedback while said natural language routine is processing said user natural language query.
- 17. The system of claim 15, wherein the interactive electronic agent is incorporated as part of an e-commerce Web page and presented within a Web browser program operating on a client device operated by the user.
- 18. The system of claim 17 wherein the electronic interactive agent is an animated character on a screen of the client device.

41

- 19. The system of claim 15 further including a routine to configure perception related parameters of the electronic interactive agent, including one of a gender, a visual appearance, and/or voice characteristics including one of pitch, volume and/or speed.
- 20. The system of claim 15, wherein at least one of said prompt, confirmation or response is processed by a text to speech engine and rendered into audible form for the user by the interactive electronic agent.
- 21. The system of claim 15, wherein the interactive 10 electronic agent automatically communicates in a voice with speech characteristics based on a native language used by the user
- 22. The system of claim 15 wherein said subset of words and/or phrases in said natural language query can be 15 assigned different weightings determined by said natural language routine.
- 23. The system of claim 15, wherein said speech recognition engine is distributed between a client device and a server system which receives streaming speech data having 20 reduced latency data content before silence is detected and the utterance is complete.
- **24**. The system of claim **15**, wherein said response is selected based on comparing which of said subset of words and/or phrases in said speech-based query also appear in 25 said query/answer pairs.
- 25. The system of claim 15, wherein a single response is selected based on determining which of said query/answer pairs has an optimal number of words and/or phrases overlapping with said natural language query.
- 26. The system of claim 15 further including providing a second natural language routine at a separate computing system adapted to assist with a response to said natural language query, such that said natural language query is recognized by said speech recognition engine, and multiple 35 databases at different server systems can be considered in responding to natural language queries for a set of topic entries.
- 27. A method of interacting with a user of a client-server based natural language query system comprising the steps 40 of:
 - a) initiating a first interactive speech session between a client device and a server system on behalf of a user;
 - b) providing a database of query/answer pairs coupled to the server system concerning one or more topics which 45 can be responded to by the natural language query system during said first interactive speech based session with said user;
 - c) providing a speech recognition engine distributed between the client device and server system, and such

42

that speech data is configured for said first interactive speech session to have a reduced data content to reduce latency, and is streamed to the server before silence is detected;

- wherein said speech recognition engine is adapted to recognize a first set of words and/or phrases received from said client device from a user during said first interactive speech session concerning a first topic taken from said one or more topics;
- further wherein said first set of words and/or phrases can include a natural language query presented as continuous natural language spoken data;
- c) providing a natural language routine coupled to the server system adapted to process said first set of words and/or phrases and identify a response to said natural language query based on said query/answer pairs;
 - wherein said natural language routine is adapted to consider only a subset of said first set of word and/or phrases, and further can consider words and/or phrases in said natural language query which are not present in said query/answer pairs to determine said response;
- d) providing an interactive electronic agent coupled to said natural language routine which is customized for the user and configured to:
 - provide a prompt to the user during said first interactive speech based session with suggestions on queries which can be made to the natural language query system concerning said first topic;
 - ii. provide a confirmation of a substance of said natural language query;
 - iii. provide said response to the user from the natural language query routine;
- wherein said interactive electronic control agent is configured to provide iterative audible responses to user natural language queries directed to said one or more topics until said interactive speech based session is terminated.
- 28. The method of claim 27 further including a step: customizing perception related parameters of the electronic interactive agent, including one of a gender, a visual appearance, and/or voice characteristics including one of pitch, volume and/or speed.
- 29. The method of claim 27, wherein the electronic interactive agent can conduct a second interactive speech based session with said user directed to continue with additional queries and answers concerning said first topic.

* * * * *

KENT DECLARATION EXHIBIT 5

'846 Claim	Accused NSRS Features ¹	Prior Art NSRS Features
11. A distributed voice recognition system comprising:	Wells Fargo employs at least three separate interactive voice response (IVR) systems found to date at 800-642-4720 (banking); 800-368-7550 (Wells Fargo Advantage Funds) and 800-TRADERS (872-3377) (Wells Fargo Investments). While testing has only been done extensively on the banking voice response system, the other systems are believed to perform the steps below as well. See e.g. PHO10434 – 10438 ("Speak your responses and the Automated Phone System will take care of the rest") and PHO10440 (tell the Automated Voice Response System what you would like to do") WF's system is distributed because some of the speech processing is done at one computing device (a speech board and/or at a speech recognition client), and the remainder of the speech processing is then completed at another computing device (a speech recognition server). [PHO004156 - 4159]	Nuance NSRS version 6 used a client/server architecture to recognize speech. Nuance's website described the architecture as follows: "Powerful client/server software architecture Nuance's distributed architecture allows recognition to happen on a separate stand-alone machine or network of servers, expediting the recognition process and resulting in memory savings because recognition data objects can be shared simultaneously across multiple recognition resources by dynamically allocating recognition resources with the Resource Manager, Nuance6 capitalizes on the efficiency of processing of multiple recognition requests in parallel and provides automatic fail-over protection in the event of a hardware or software failure. Nuance's client/server architecture also allows applications developers to decouple audio acquisition from recognition processing. The RecClient performs all audio and telephony functions required by speech applications, including echo cancellation and endpointing when DSP is unavailable, and manages the collection and transmission of speech to the multi-threaded RecServer. The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user [Kent Decl., Ex. 10 (Nuance6), at WF00000595]. The Developer's Manual describes the recognition client as performing endpointing: "The RecClient then attaches to the audio device for the application-for example, a telephone port-and provides the application with the following functionality: - Energy-based endpointing to determine the beginning and end of the talker's speech

_

¹ The accused NSRS features are quoted directly from Phoenix's Infringement Contentions and have been highlighted to make the comparison of key elements easier. Where Phoenix has accused multiple elements in the NSRS, the chart includes one or more of the elements that Phoenix contends are sufficient to prove infringement.

		Nuance recognition servers."
		[Sharp Ex. A (Nuance Developer's Manual version 6), at WF00000717].
		The Developer's Manual explains that the recognition client and recognition server can be run on different machines:
		"Because a speech recognition application actually performs recognition only a certain percentage of the time, the RecServer was designed to be used intermittently by multiple RecClients running different applications. This allows the CPU and memory resources of the RecServer to be shared across multiple client applications, and provides the following advantages:
		- Distributed Architecture. Recognition, which can be CPU-intensive, can be run on a different machine from the machine running the application and audio interface.
		[Id. at WF00000724].
		Further, the recognition client was intended to be operated on, or in conjunction with, telephone interfaces such as a Dialogic telephony board:
		"The Nuance System RecClient supports several telephone interfaces, including Dialogic."
		[<i>Id.</i> at WF00000774].
		The Dialogic telephony board was capable of performing echo-cancellation and endpointing and passing that data to the recognition server:
		"Consistent with the Nuance architecture, echo- canceled, endpointed audio data are recognized by the Nuance Recognition Server, running on the same host or another host."
		[Id., at WF00000903; see also WF00000896, WF00000899-904 (describing Nuance Audio And Telephony Interfaces including Dialogic Antares card that handles endpointing and echocancellation)].
		Sharp Decl., ¶ 9, 17.
11.1 a sound processing circuit adapted to receive a speech utterance and	One component of the WF IVR hardware is a speech board with a processing circuit which processes the user utterances received over a phone/network line. This is a telephony board or some other signal	The NSRS version 6 was intended to be used with Dialogic boards. These boards received voice signals coming from connected telephone/network lines, digitized those signals, and then passed along

to generate associated speech utterance signals therefrom; processing circuit board within a computing system. [PHO004155 – 57; 3007] Examples of the type of speech boards which must be employed by WF for such purpose are illustrated in the discovery materials provided by Phoenix. [PHO004313 – 4317; 4249 – 4260; 2739 -2742; 4318-4332; 4577 – 4578; 4490 – 4494] The Nuance 8.x datasheet specifically identifies many of these boards. [PHO003007]

the live audio data to the recognition client:

"An audio provider, a set of C interfaces that handle the reception and transmission of live audio data to and from the current hardware. Audio providers are described in Chapter 15, "Nuance Audio and Telephony Providers."

[Sharp Ex. A (Nuance Developer's Manual version 6) at WF00000719].

"The recognition client provided with the Nuance System processes live audio data read from one of the supported audio devices (see Chapter 15 for more information). With audio access taken care of, you are free to concentrate on the workings of the application, without having to deal with the routine transfer of digitized audio from the audio device to the server. For applications that need to recognize audio originating from a source that Nuance does not support, you may need to implement a custom audio provider (see Chapter 15). However, most developers use the RecClient API with the existing audio providers to write their speech recognition applications."

[Id. at WF00000762].

The Nuance system worked with Dialogic interfaces:

"The Nuance System RecClient supports several telephone interfaces, including Dialogic."

[Sharp Ex. A (Nuance Developer's Manual version 6), at WF00000774; *see also id.* at WF00000899-904 (describing Dialogic as an appropriate Nuance Audio And Telephony Interfaces)].

See Sharp Decl., \P 13.

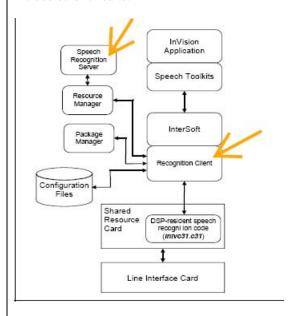
11.2 and a first signal processing circuit adapted to generate a first set of speech data values from said speech utterance signals, said first set of speech data values being insufficient by themselves for permitting recognition of words articulated in said speech utterance;

The speech board analyzes the utterances from the speaker and participates in the speech recognition process. One of the things it can do, when so configured by WF, is look for whether there is actually speech or not from the user (as opposed to silence, or some other background noise). When this feature is enabled it is implemented by a voice activity detector (VAD) circuit. [PHO004313 – 4317; 4249 – 4260; 2739 -2742; 4318-4332; 4577 – 4578; 4490 – 4494; 3915 - 3924] Other speech processing operations may also be enabled and performed. [PHO003796; 3842; 3888 – 3901]

The NSRS version 6 used a client/server model wherein the recognition client could look for whether there is actual speech or not from the user (i.e., endpointing):

"Nuance6 capitalizes on the efficiency of processing of multiple recognition requests in parallel and provides automatic fail-over protection in the event of a hardware or software failure Nuance's client/server architecture also allows applications developers to decouple audio acquisition from recognition processing. The RecClient performs all audio and telephony

This limitation is also separately met by a WF recognition client (which runs on a computing system which the telephone board can be a part of) that can also implement some of these functions. [PHO4254] See e.g. PHO PHO002900-04 "... The Speech Integration Module—Nuance speech recognition system uses a client-server architecture. The client consists of one or more IVRs, containing all the telephony components used to accept and control a connection from an end user. ... speech recognition begins on the shared resource boards in the IVR. ..." The WF SR client and WF SR server are shown generally with the arrows above but can include other circuits.



The data speech data values generated by the speech board (or recognition client) are not sufficiently processed to allow for correct recognition of words. They must be processed by a second circuit as discussed below.

A transmission circuit (located on the speech board or part of a recognition client) sends the first set of speech data values over a channel (i.e., a network connection or the like) to a second circuit, in this

functions required by speech applications, including echo cancellation and endpointing when DSD is unavailable, and manages the

when DSP is unavailable, and manages the collection and transmission of speech to the multi-threaded RecServer. The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user"

[Kent Decl., Ex. 10 (Nuance6), at WF00000595].

"The RecClient performs audio and telephony functions, freeing applications to focus on higher-level issues such as call flow and dialog management. . . . The RecClient then attaches to the audio device for the application-for example, a telephone port-and provides the application with the following functionality: . . .

- Energy-based endpointing to determine the beginning and end of the talker's speech
- Recognizing speech via dynamic connections to Nuance recognition servers."

see also [Sharp Ex. A (Nuance Developer's Manual version 6) at WF00000717].

Further, the Nuance system worked in conjunction with telephony boards that also could perform echo-cancellation and endpointing:

"The Nuance System RecClient supports several telephone interfaces, including Dialogic."

[*Id.* at WF00000774];

"Consistent with the Nuance architecture, echocanceled, endpointed audio data are recognized by the Nuance Recognition Server, running on the same host or another host."

[*Id.* at WF00000903; *see also* WF00000896, WF00000899-904 (describing Nuance Audio And Telephony Interfaces including Dialogic Antares card that handles endpointing and echocancellation)].

Sharp Decl., \P 9.

As discussed above, the NSRS version 6 included a recognition client and was capable of being used with a telephony board such as those made by Dialogic.

11.3 and a

transmission circuit

transmitting said first

for formatting and

set of speech data values over a communications channel to a second signal processing circuit; 11.4 wherein said first set of speech data values are sent in a streaming fashion over said channel before silence is detected and/or said speech utterance is completed;	case, the speech recognition server. [PHO004313 – 4317; 4249 – 4260; 2739 -2742; 4318-4332; 4577 – 4578; 4490 – 4494; 3915 – 3924; 3796; 3842; 3888 – 3901] See above citations which explain how the data is sent this way (streaming) by the speech recognition board (or by the recognition client as the case may be) before silence is detected, and/or while the user is still talking, to expedite the SR processing.	Either the recognition client and/or the telephony board would send the processed speech data over a network connection to the recognition server: "The processed speech values are then sent over a network connection to the RecServer which handles the speech recognition and understanding." [Kent Decl., Ex. 10 (Nuance6), at WF00000595]. "Samples begin flowing to the recserver as soon as the onset of speech is detected, and the flow continues until the end of speech is found. In this way, the recserver can actually begin to process the utterance while the person is still speaking." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000968]; see also id. at WF00000727-728 (describing connections between RecClient and RecServer). To send the processed data, there was a circuit to format and transmit the speech values to the RecServer. Sharp Decl., ¶ 14. The endpointed data output from the recognition client was sent to the recognition before silence is detected and/or the speech utterance is completed: "Samples begin flowing to the recserver as soon as the onset of speech is detected, and the flow continues until the end of speech is found. In this way, the recserver can actually begin to process the utterance while the person is still speaking." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000968]. Sharp Decl., ¶ 14.
11.5 and said second signal processing circuit being configured to generate a second set of speech data values based on receiving and processing said speech data values during said speech utterance and before silence is detected, such that second set of speech	The main SR server includes hardware and software which act as the second signal processing circuit. [PHO004313 – 4317; 4249 – 4260; 2739 - 2742; 4318-4332; 4577 – 4578; 4490 – 4494; 3915 – 3924; 3796; 3842; 3888 – 3901] A set of audio vectors (second set of speech data valued) are derived from the first set of speech data values. These audio vector data are what is needed by the speech recognizer to actually recognize words in the speech utterance.	The NSRS version 6 included a recognition server that completed speech recognition using what Phoenix calls "audio vectors": "The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user" [Kent Decl., Ex. 10 (Nuance6) at WF00000595]. That server derived a sequence of phonetic

sufficient information	
	recognition and to recognize words in the speech
to be usable by a word	utterance:
recognition engine for	
recognizing words in	"The Nuance Speech Recognition System
said speech utterance;	represents each word in the recognition
	vocabulary as a sequence of phonetic symbols.
	These symbols provide the correspondence
	between a word in the dictionary and its
	pronunciation."
	[Sharp Decl., Ex. A (Nuance Developer's Manual
	version 6) at WF00000988].
	"To recognize speech, the Nuance System
	compares it with a graph of acoustic speech
	models. In a process called the recognition search,
	the recognition engine searches this graph of
	possibilities for the sequence of models that best
	correspond to the speech. These models are found
	and interpreted as strings of words, and
	returned as the recognition result."
	[<i>Id.</i> at WF00000654].
	"Madala and he annual according to their
	"Models can be grouped according to their
	underlying technologies:
	Genone models—whose names start with "gen"—
	are the Nuance System's most accurate continuous-
	density HMMs.
	delisity Hivilvis.
	Phonetically tied mixture (PTM) models are also
	continuous-density HMMs, but PTM models use
	fewer Gaussians than Genone models, and are
	generally faster but slightly less accurate.
	generally ruster out slightly less decurate.
	Vector-quantized (VQ) models use discrete-density
	probability functions. VQ models can be the fastest
	models that Nuance Communications ships, but
	they tend to make more errors than either PTM or
	Genone models.
	[Sharp Decl., Ex. A (Nuance Developer's Manual
	version 6) at WF00000657]; see also d. at
	WF00000967) (describing the acoustic speech

models).

recognize speech:

The Nuance recognition server used HMMs to

"Naturally, the Nuance System uses state-of-theart HMMs at the core of its recognition engine.

HMMs function primarily as acoustic models

11.6 and further wherein at least some words are recognized in real-time and output as text before said speech utterance is completed.	The Phoenix protocols and audio files demonstrate the real-time behavior of the WF SR system in recognizing words. [PHO006352 – 6368; 6377 – 6380] This limitation is separately met for those systems in which WF configures the IVR to operate using partial results [PHO003165; 3212] which is also characterized by the words being recognized before the utterance is completed.	that provide a mapping from the sampled speech signal to a sequence of phonetic units." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000964; see also id. at WF00000965-968 (describing HMM)]. The Nuance recognition server would return recognized words: "The RecResult structure contains the final recognition result, the NLResult, and several recognition result, the NLResult, and several recognition statistics, including the confidence score of the recognition result. If N-best recognition was requested (by running the server with the parameter), the RecResult structure may contain more than one recognition hypothesis. See the online documentation , and associated functions for details on extracting recognition strings and other information from the RecResult structure." [Id. at WF00000769]. Sharp Decl., ¶ 17. The NSRS version 6 also operated in real time. Indeed, "[s]amples begin flowing to the recserver as soon as the onset of speech is detected, and the flow continues until the end of speech is found. In this way, the recserver can actually begin to process the utterance while the person is still speaking." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000968]. Further, the NSRS version 6 was capable of returning partial results: "Set this Boolean parameter to TRUE if you want the recognition engine to generate partial (intermediate) recognition results periodically. Each result will generate a NUANCE-EVENT-PARTIAL-RESULT." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000878]. Sharp Decl., ¶ 18.
17. The system of claim 11, wherein signal processing	This limitation is met by multiple facets of WF's IVR system(s), including one or more of the following:	The NSRS version 6 system was compatible with the Dialogic Antares platform. [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at

functions required to generate said first and second set of speech data values can be allocated between said first signal processing circuit and second signal processing circuit as needed based on computing resources available to said first and second signal processing circuits respectively.

- 1) As noted above the voice activity function can be selectively enabled depending on whether such resources are available on the particular type of speech board. [PHO004313 4317; 4249 4260; 2739 -2742; 4318-4332; 4577 4578; 4490 4494; 3915 3924]
- 2) Other speech processing operations performed by the recognition client, such as endpointing may also be selectively enabled and performed there or on the SR server. [PHO003140 3150]
- 3) For any WF IVR system using an SLM grammar, the degree of pruning of the SLM grammar by the server (second signal processing circuit) can be controlled [PHO003352]

WF00000899-901].

Such boards were capable of being configured to handle more or less signal processing functions:

"Rather than offering a specific high level product, DSPSE offers the means to create custom products based on DSP technology with the Antares platform. DSPSE has 21 algorithm products that are compatible with the Antares platform and which can be used to quickly build high performance applications that fit specific customer requirements. The Systems Group at DSPSE has used these algorithm products to build "private label" solutions on the Antares platform. The combination of Antares specific experience and expertise in the development of DSP algorithms and applications enables DSPSE to quickly create the exact DSP based solution your application requires."

[Kent Decl., Ex. 15 (Antares Card)].

Further, the NSRS version 6 discloses adjustments that can be made to endpointing at pages 21-22.

In addition, a developer can disable endpointing and simply specify the start and end of speech:

"The RecClient API also allows the developer to specify the start and end of the utterance to be recognized, bypassing the spectral energy endpointing used by RCRecognize ().

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000770].

If endpointing was disabled, all samples were sent to the recognition server:

"The ability to perform endpointing allows more computationally efficient use of a recognition server's resources. When endpointing is not performed, all samples received by the client process are sent to the recognition server. Because users often pause after they are prompted to speak, some silence samples are unnecessarily sent to the recognition server. This increases the load put on the recognition server and lengthens the delay before the recognition server can respond."

[Id. at WF00000968].

Further, the NSRS version 6 permitted the configuration of the degree of pruning:

"The accuracy and speed of the Nuance System is determined by several factors: . . .

The amount of search performed. The recognition system can be configured to perform less search. This will speed up the system but might cause search errors, if good theories are missed.

The performance of the Nuance System can be optimized by varying system configurations.

[Id. at WF00000654-655].

"rec.Pruning

This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased."

[Id. at WF00000657].

"rec.PPR

This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a smallvocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for largevocabulary tasks, and FALSE for small-vocabulary tasks."

Id, at WF00000658.

"rec.Pruning

Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."

		[Id. at WF00000877].
		Sharp Decl., ¶ 19.
20. The system of claim 11, wherein said first signal processing circuit is also configured to assist said second signal processing circuit with signal processing computations required to generate said second set of speech data values.	As seen in the materials noted above, the speech boards and/or recognition client can perform an endpointing function on the speech data from the user utterance. [PHO004313 – 4317; 4249 – 4260; 2739 -2742; 4318-4332; 4577 – 4578; 4490 – 4494; 3915 - 3924] Endpointing is used to determine when a user has stopped speaking. In PHO003140 – 3150 it can be further seen that the SR engine used by WF can be configured so that recognition server also performs this specific function of speech endpointing. Thus, in this instance, the SR board (and/or the recognition client) assists the recognition server with signal processing computations.	In the NSRS version 6, the Recognition Client was able to perform an endpointing function on the speech data from the user utterance. "The RecClient then attaches to the audio device for the application-for example, a telephone port-and provides the application with the following functionality: - Energy-based endpointing to determine the beginning and end of the talker's speech - Recognizing speech via dynamic connections to Nuance recognition servers." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000717; see also id. at WF00000660-661]. In addition, the Nuance-compatible telephony board (e.g., the Dialogic telephone interface) was able to perform an endpointing function: "The Nuance System RecClient supports several telephone interfaces, including Dialogic." [Id. at WF00000774]; "Consistent with the Nuance architecture, echocanceled, endpointed audio data are recognized by the Nuance Recognition Server, running on the same host or another host." [Id. at WF00000903; see also id. at WF00000896, WF00000899-904 (describing Nuance Audio And Telephony Interfaces including Dialogic Antares card that handles endpointing and echocancellation)]. Sharp Decl., ¶ 9.
21. The system of	As seen in the materials noted above, with VAD	The NSRS version 6 offered the ability to perform
claim 11, wherein said first set of speech data values represent the least amount of data	enabled, the speech boards and/or recognition client transmit only actual speech data from the user utterance to the SR server, which represents the least amount of data presented in the user	endpointing which sends only speech data (and not silence) to the recognition server: Indeed, "[w]hen endpointing is not performed, all

that can used by said	utterance. [PHO004313 – 4317; 4249 – 4260; 2739 -	samples received by the client process are sent to
second signal	2742; 4318-4332; 4577 – 4578; 4490 – 4494; 3915 -	the recognition server."
processing circuit to	3924]	
generate said second		[Sharp Decl., Ex. A (Nuance Developer's Manual
set of data values		version 6) at WF00000968].
usable for a word		
recognition process.		Accordingly, endpointing is used to prevent the following:
		"Because users often pause after they are prompted to speak, some silence samples are unnecessarily sent to the recognition server. This increases the load put on the recognition server and lengthens the delay before the recognition server can respond."
		[<i>Id</i>].
		Thus, when endpointing is used, only the speech that is detected is transmitted to the SR server:
		Samples begin flowing to the recserver as soon as the onset of speech is detected, and the flow continues until the end of speech is found. In this way, the recserver can actually begin to process the utterance while the person is still speaking.
		[<i>Id</i>].
		This is also reflected by fact that "[s]ometimes the word that the user says may not have enough energy in the beginning of the word to set off the endpointer. As a result, only the middle portion of the word is sent to the recognizer."
		[Id. at WF00000969].
		Sharp Decl., ¶ 10.

KENT DECLARATION EXHIBIT 6

'640 Claim	Accused NSRS Features ¹	Prior Art NSRS Features
1. An interactive learning system adapted for responding to speech-based queries concerning topics addressed by such interactive learning system, the system comprising:	The Wells Fargo interactive voice response (IVR) systems are interactive learning systems (hardware and software) that respond to speech –based queries from WF customers concerning certain banking and investment topics. WF employs at least three separate IVR systems found to date at 800-642-4720 (banking); 800-368-7550 (Wells Fargo Advantage Funds) and 800-TRADERS (872-3377) (Wells Fargo Investments). While testing has only been done extensively on the banking voice response system, the other systems are believed to perform the steps below as well. See e.g. PHO10434 – 10438 ("Speak your responses and the Automated Phone System will take care of the rest") and PHO10440 (tell the Automated Voice Response System what you would like to do") WF's IVR systems are adapted for helping educate/inform customers how to transfer funds, learn their bank account balance, get stock quotes, etc.	Prior art NSRS was implemented in IVR systems to respond to speech –based queries from customers concerning certain banking and investment topics. For example, Schwab implemented VoiceBroker using NSRS "to provide stock, mutual fund and market indicator information to retail customers." [Kent Decl., Ex. 9 (1996 VoiceBroker Press Release) at WF00000593-594]. Sample IVR implementations of NSRS version 6 responded to speech-based queries regarding banking, travel, and stock quote information. [Kent Decl., Ex. 10 (Nuance6) at WF00000597-604]. The Nuance Developer's Manual version 6 also provides a "a sample banking application built with the Dialog Builder. This application lets you check your balance, transfer money, pay bills, and add new payees. The "add payee" feature illustrates the enrollment and dynamic grammar capabilities of the Nuance software. Everything you need to build and run the application is provided, including the source code, grammar, dictionary, prompts, Makefile, and scripts." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000956].
1.1 a query file for storing a plurality of topic query entries, each topic query entry including a query relating to one or more of the topics covered by the speech-based interactive learning system;	WF's systems include a query file (an electronic data structure stored on a computing system) which stores entries relating to the topics the system can cover with customers. For example, as seen in [PHO010425 – 432] one of the topics the IVR covers is account histories (other topics are also identified at the main menu).	Implementations of NSRS version 6 covered topics such as "account balance" among other topics such as "transfer money" and "pay bills." [Kent Decl., Ex. 10 (Nuance6) at WF00000599-600]. The Nuance Developer's Manual version 6 also provides a "a sample banking application built with the Dialog Builder. This application lets you check your balance, transfer money, pay bills, and add new payees. The "add payee" feature illustrates the enrollment and dynamic grammar capabilities of the Nuance software. Everything you need to build and run the application is provided,

_

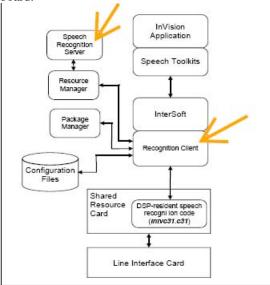
¹ The accused NSRS features are quoted directly from Phoenix's Infringement Contentions and have been highlighted to make the comparison of key elements easier. Where Phoenix has accused multiple elements in the NSRS, the chart includes one or more of the elements that Phoenix contends are sufficient to prove infringement.

		including the source code, grammar, dictionary, prompts, Makefile, and scripts." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000956; see also id. at WF00000686-687, WF00000739, WF00001012-1022]. See Sharp Decl., ¶ 27.
1.2 an answer file for storing a plurality of topic answer entries, each topic answer entry including an answer to one or more of said plurality of topic query entries, such that each topic query entry has at least one associated topic answer entry;	Again, see PHO0006352 – 6368; 4473] the WF IVR knows to give an appropriate answer when it figures out what the query/topic is all about. These answers are also stored within an electronic data structure, so that for each topic the IVR covers, it can give the customer at least one answer. For example, on PHO010432 the IVR asks for what service the customer is calling about	Implementations of NSRS version 6 knew how to give an appropriate answer when the system figured out what the query/topic is all about. The NSRS version 6 depicted in the Banking demonstration knew to give an appropriate answer when the system figured out what the query/topic is all about. This is shown on page 100 of the Developer's Manual. Moreover, the demonstrations of the NSRS version 6 knew how to give an appropriate answer when the system figured out what the query/topic is all about. For example, the following review explains that: "The few tests we tried here gave us favorable responses. When asked how much we wanted to transfer from checking into savings, we responded, 'Fifty-seven dollars and thirty-two cents.' Later, in response to the same question, we said, 'Fifty-seven thirty-two.' In both cases, the demo transferred the correct amount." "We started by referring to an airport first as JFK, then as Kennedy. The system responded appropriately, recognizing that both names corresponded to the same airport. So, we decided to try something a little trickier. We decided to refer to Dulles, for Dulles International Airport. We supposed the system might mistake Dulles for Dallas, which sounds much the same. We supposed wrong, however. The system did in fact recognize Dulles, and we had to admit we were impressed." See [Kent Decl., Ex. 11 (Oct. 1998 Review)]. Further information is provided in Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000686-687, WF00000739, WF00000956, WF00001012-1022. See Sharp Decl., ¶ 27.
1.3 a speech recognition system for	The Wells Fargo IVRs provide have a <i>speech</i> recognition (SR) engine that meets this limitation as	NSRS version 6 included a recognition client that processed the input speech signals by (among other

generating recognized speech utterance data from partially processed speech data associated with a speech-based query concerning one of said topics, said partially processed speech data being received from a remote speech capturing system; acknowledged in WF materials (see PHO10434: "..using advanced voice recognition...") and the literature of the various vendors (including Intervoice, Nuance) who supplied the components of the IVR. (PHO010442-51; PHO010464 (Intervoice identified as source of Omvia speech enabled system); PHO010469; PHO010619-626 (Nuance further identified as assisting with system).

In the system used by WF, partially processed speech data is sent by a speech board and/or recognition client, which are located in another computer separate (remote) from the speech recognition server. [PHO004156]

See e.g. PHO PHO002900-04 showing the SR server is remote from the recognition client and speech board.



The data speech data values generated by the speech board (or recognition client) are only partially processed and not sufficient to allow for correct recognition of words. They must be processed by the SR server as seen in the figure above.

things) performing echo cancellation and endpointing:

"Nuance6 capitalizes on the efficiency of processing of multiple recognition requests in parallel and provides automatic fail-over protection in the event of a hardware or software failure Nuance's client/server architecture also allows applications developers to decouple audio acquisition from recognition processing. The RecClient performs all audio and telephony functions required by speech applications, including echo cancellation and endpointing when DSP is unavailable, and manages the collection and transmission of speech to the multithreaded RecServer. The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user"

[Kent Decl., Ex. 10 (Nuance6) at WF00000595].

"The RecClient then attaches to the audio device for the application-for example, a telephone port-and provides the application with the following functionality: . . .

- Energy-based endpointing to determine the beginning and end of the talker's speech
- Recognizing speech via dynamic connections to Nuance recognition servers."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000717].

The recognition client was able to be run on a machine "remote" from the RecServer:

- "Because a speech recognition application actually performs recognition only a certain percentage of the time, the RecServer was designed to be used intermittently by multiple RecClients running different applications. This allows the CPU and memory resources of the RecServer to be shared across multiple client applications, and provides the following advantages:
- Distributed Architecture. Recognition, which can be CPU-intensive, can be run on a different machine from the machine running the application and audio interface.

[Id. at WF00000724].

NSRS version 6 also included the capability of using a Dialogic telephony board that could process the input speech signals by (among other things) performing echo cancellation and endpointing:

"The Nuance System RecClient supports several telephone interfaces, including Dialogic."

[Id. at WF00000774].

"Consistent with the Nuance architecture, echocanceled, endpointed audio data are recognized by the Nuance Recognition Server, running on the same host or another host."

[*Id.* at WF00000903; *see also id.* at WF00000896, WF00000899-904 (describing Nuance Audio And Telephony Interfaces including Dialogic Antares card that handles endpointing and echocancellation)].

The data output from the RecClient was then passed to the RecServer which generated recognized words:

"The processed speech values are then sent over a network connection to the RecServer which handles the speech recognition and understanding."

[Kent Decl., Ex. 10 (Nuance6) at WF00000595].

"Samples begin flowing to the recserver as soon as the onset of speech is detected, and the flow continues until the end of speech is found. In this way, the recserver can actually begin to process the utterance while the person is still speaking."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000968].

"To recognize speech, the Nuance System compares it with a graph of acoustic speech models. In a process called the recognition search, the recognition engine searches this graph of possibilities for the sequence of models that best correspond to the speech. These models are found and interpreted as strings of words, and returned as the recognition result."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000654; *see also id.* at WF00000964-968 (describing HMM)].

"The RecResult structure contains the final recognition result, the NLResult, and several

1.4 said speech recognition system further cooperating with a natural language engine, which processes said recognized speech utterance data using a morphological analysis and a phrase analysis to form recognized speech

sentence data

corresponding to said

speech-based query;

The Wells Fargo IVRs provide a *natural language* (NL) engine that meets this limitation as acknowledged in the literature of the various vendors who supplied the components of the IVR. See e.g. PHO002806 ".. The Nuance System lets you define natural language understanding in your recognition packages. A natural language understanding system takes a sentence (typically a recognized utterance) as input and returns an interpretation—a representation of the meaning of the sentence." See also PHO0002807-2821.

This routine is used to determine the meaning of the user's query based on the words/phrases identified by the SR engine. This NL engine is also embodied in software, and typically operates on the speech server as well. In the case of the WF IVR this engine is embodied either as part of the Nuance 7.x/8.x software....

Phrase analysis: At least a part of this NL engine takes the recognized words from the speech recognizer and performs a linguistic analysis (i.e, grouping of words) to develop/identify speech sentence data corresponding to the user's query, such as "account balance."

The WF IVR's NL engine behavior is shown in the testing protocols provided by Phoenix [PHO10425 - 10433] which demonstrate that the IVR can understand phrases such as "account balance." The fact that IVR can identify word phrases is apparent from the audio clips also provided by Phoenix. [PHO10423 -10424]

Morphology analysis: The audio clips further illustrate that the IVR understands different morphologies of words ("I need help transferring funds" gets understood as "transfer" funds. The selected word phrases which the NL engine looks for are embodied as part of a natural language grammar which WF again has customized for its particular

recognition statistics, including the confidence score of the recognition result. If N-best recognition was requested (by running the server with the parameter . . .), the RecResult structure may contain more than one recognition hypothesis. See the online documentation . . . , and associated functions for details on extracting recognition strings and other information from the RecResult structure."

[Id. at WF00000769].

See Sharp Decl., ¶ 9, 11, 17.

The NSRS version 6 included, when implemented, what Phoenix contends is a "natural language engine" that took a sentence (typically a recognized utterance) as input and returns an interpretation—a representation of the meaning of the sentence:

"The Nuance System allows developers to quickly and easily create powerful natural language understanding systems tailored to their needs. A natural language understanding system takes a sentence as input and returns as output an interpretation, which is a representation of the meaning of the sentence..."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000678].

The NSRS version 6 was able to understand phrases such as "transfer five hundred dollars from savings to checking":

"The interpretation for the command "Transfer five hundred dollars from savings to checking" would then probably fill the command-type slot with transfer, the source-account slot with savings, the destination-account slot with checking, and the amount slot with 500."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF0000678; *see also id.* at WF00000679-686].

The NSRS version 6 can understand different forms of a word such as "Give me" and "Gimme":

"For example, users will often say "Give me the cheapest flight" as "Gimme the cheapest flight." You can improve accuracy by explicitly including these compound words with additional paths in your grammar, for example:

([(give me) give-me] the cheapest flight)

domain – i.e. its business/customer relationship management operations. [PHO003320 - 3361; 4473]

and augmenting your dictionary, for example:

give-me g ih m iy

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000661-662].

Further, the NSRS version 6 includes a templatematching natural language component that, when implemented, can understand phrases and different morphologies of words:

"The Nuance System allows developers to quickly and easily create powerful natural language understanding systems tailored to their needs. A natural language understanding system takes a sentence as input and returns as output an interpretation, which is a representation of the meaning of the sentence. . . .

Natural language understanding applications built with the Nuance System produce interpretations consisting of slots and values for these slots. For any given application, the developer specifies a fixed set of slots, which correspond to types of information commonly supplied in queries in the application's domain. For example, in an automated banking application, slots might include command-type, amount, source-account, and destination account.

The interpretation for the command "Transfer five hundred dollars from savings to checking" would then probably fill the command-type slot with transfer, the source-account slot with savings, the destination-account slot with checking, and the amount slot with 500."

[*Id.* at WF00000678; *see also id.* at WF00000679-686].

"Once the RecServer has recognized the speech, it uses a template-matching natural language approach to determine meaning. This means the RecServer takes the relevant pieces of recognized speech and matches them against a customized template of prespecified information. This template approach is more robust and sophisticated than word spotting or "tag" approaches and easier and more efficient than an application-specific sentence parser."

[Kent Decl., Ex. 10 (Nuance6) at WF00000596].

See Sharp Decl., ¶ 21, 22.

1.5 a query formulation system for converting said recognized speech sentence data into a search query suitable for identifying a topic query entry corresponding to said speech-based query, and for locating at least one topic answer entry best matching said speech-based query.

Query formulation system - The IVR also includes a query formulation (QF) system. This QF system is embodied in software which typically operates on the speech recognition server as well and may be implemented as part of the SR and NL engines as well. The QF system identifies the topic corresponding to the query, and locates an answer best matching the query. A match might be as simple as routing the user to a particular agent with expertise in the area of interest gleaned from the query. Alternatively, a match might include a response provided by the IVR to the user, such as the account history.

That this system exists is again demonstrated from the Phoenix protocols which show the system retrieving answers (such as an account balance) in response to determining the caller's intent.

determining the caller's intent as seen in the Nuance demonstrations.

[Kent Decl., Ex. 10 (Nuance6) at WF00000597-604]; [Kent Decl., Ex. 11 (Oct. 1998 Review)].

In NSRS version 6, the system retrieved answers

(such as an account balance) in response to

One of these examples is the stock quote demonstration that "shows Nuance's ability to provide highly accurate quotes on over 13,000 stocks, mutual funds, and market indicators. To reach the demo, call 650-847-7423. Simply say the name of the company, fund, or market indicator."

[Kent Decl., Ex. 10 (Nuance6) at WF00000603].

Indeed, NSRS version 6 generated a database query to retrieve answers (such as an account balance) in response to determining the caller's intent:

"Two database systems are currently supported: Informix and Mini SQL. These are both relational database systems that process SQL queries. . . .

If you pass the argument -db database-name into AppNew (), the system will automatically connect to the given database (and disconnect when the . . .object is deleted).

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000759].

"travelT.h is application handles requests for flight information, much like a human travel agent would. It is very useful to look at this application while you are learning about SpokenSQL. The examples in this chapter will also be in the flight information domain."

[*Id.* at WF00000935].

"This function generates the query from the active interpretation of the given NLResult object. The query produced is placed in the query buffer, provided that it fits."

[*Id* at WF00000935].

A simple example is explained with respect to a flight table in pages WF00000937-WF00000938, WF00000939-WF00000955.

See Sharp Decl., ¶ 26.

3. The system of claim 1, wherein said speech recognition system is comprised of a first portion at a client based computing system for performing first signal processing operations on a speech input signal to create said partially processed speech data, and a second portion at a server based computing system for performing a second signal processing operation for completing processing of said partially processed speech data.

In the SR system, a first portion component consists of the SR board and/or the speech recognition client which create the partially processed speech data. See above discussion for SR boards and [PHO004155 – 4159] As noted above, the second portion component of the SR system includes the SR server which performs additional operations to complete the processing of the speech data.

NSRS version 6 used a client/server architecture where the client portion is the recognition that performed endpointing and the server portion is the recognition server which performed additional operations to complete the processing of the speech data:

"Nuance6 capitalizes on the efficiency of processing of multiple recognition requests in parallel and provides automatic fail-over protection in the event of a hardware or software failure Nuance's client/server architecture also allows applications developers to decouple audio acquisition from recognition processing. The RecClient performs all audio and telephony functions required by speech applications, including echo cancellation and endpointing when DSP is unavailable, and manages the collection and transmission of speech to the multi-threaded RecServer. The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user"

[Kent Decl., Ex. 10 (Nuance6) at WF00000595].

The recognition client was capable of performing endpointing before passing data to the recognition server:

"The RecClient then attaches to the audio device for the application-for example, a telephone port-and provides the application with the following functionality: . . .

- Energy-based endpointing to determine the beginning and end of the talker's speech
- Recognizing speech via dynamic connections to Nuance recognition servers."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000717].

The RecClient was able to be run on a separate machine than the RecServer:

"Because a speech recognition application actually performs recognition only a certain percentage of the time, the RecServer was designed to be used intermittently by multiple RecClients running different applications. This allows the CPU and

memory resources of the RecServer to be shared	
across multiple client applications, and provides the	
following advantages:	

- Distributed Architecture. Recognition, which can be CPU-intensive, can be run on a different machine from the machine running the application and audio interface.

[Id. at WF00000724].

NSRS version 6 also included the capability of using a Dialogic telephony board that could process the input speech signals by (among other things) performing echo cancellation and endpointing:

"The Nuance System RecClient supports several telephone interfaces, including Dialogic."

[Id. at WF00000774].

"Consistent with the Nuance architecture, echocanceled, endpointed audio data are recognized by the Nuance Recognition Server, running on the same host or another host."

[*Id.* at WF00000903; *see also id.* at WF00000896, WF00000899-904 (describing Nuance Audio And Telephony Interfaces including Dialogic Antares card that handles endpointing and echocancellation)].

The data output from the recognition client and/or telephony board was then passed to the recognition server which performs additional operations to complete the processing of the speech data:

"Samples begin flowing to the recserver as soon as the onset of speech is detected, and the flow continues until the end of speech is found. In this way, the recserver can actually begin to process the utterance while the person is still speaking."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000968].

"To recognize speech, the Nuance System compares it with a graph of acoustic speech models. In a process called the recognition search, the recognition engine searches this graph of possibilities for the sequence of models that best correspond to the speech. These models are found and interpreted as strings of words, and returned as the recognition result."

		[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000654; see also id. at WF00000964-968 (describing HMM)]. "The RecResult structure contains the final recognition result, the NLResult, and several recognition statistics, including the confidence score of the recognition result. If N-best recognition was requested (by running the server with the parameter), the RecResult structure may contain more than one recognition hypothesis. See the online documentation , and associated functions for details on extracting recognition strings and other information from the RecResult structure." [Id. at WF00000769].
		See Sharp Decl., ¶¶ 9, 17.
4. The system of claim 1, wherein said query formulation system uses context parameters for recognizing said speech-based query.	The WF IVR system knows what context the user is experiencing when the query is presented, such as, for example, where he/she is in the dialog. The user's context is used to assist the speech recognition process by helping to recognize what the user said, such as by associating a specific grammar with the specific utterance within the dialog. [PHO002902-03; 2998; 3176; 3810; 3362; 4155; 4158]	NSRS version 6 system knows what context the user is experiencing when the query is presented, such as, for example, where he/she is in the dialog. This context is used to assist the speech recognition process. The NSRS would know, for example, that when a user said "transfer today," the word "today" referred to a particular date based on the context of the utterance: "Contextualization refers to the process of updating something to reflect the context of utterance. The function ContextualizeDate () updates a date value to reflect the time at which it was uttered." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000758]. Indeed, "[t]he usability of Nuance applications is enhanced by advanced features, such as single phrase correction, that ensures users do not need to repeat an entire sentence if only part of what they said is not recognized." [Kent Decl., Ex. 16 (Nuance6 1999 Data Sheet)].
		This is also shown in the review of NSRS version 6, where the banking demonstration could understand where the user was in the dialogue. [Kent Decl., Ex. 11 (Oct. 1998 Review)].

		This was accomplished using the function "AppSetGrammar." [Id. at WF00000751]. Sharp Decl., ¶ 24.
7. The system of claim 4, wherein said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speechbased query.	See claim 4. The context parameters are used to determine whether a customer is asking about a credit card or a bank account. Depending on the context, the IVR system identifies the appropriate grammar and dictionary file to be used in understanding the query.	NSRS version 6 allowed a system to identify the appropriate grammar to be used in understanding the query depending on the context. "The Nuance System allows the dynamic creation and modification of recognition grammars from within a running application. This capability enables many new types of speech recognition applications, such as: A speed dialer, in which speaker-dependent words are associated with phone numbers-a personal speed dial list is maintained for each enrolled user of the system, and the user's own grammar of names (such as "mom", "the office", and "robert jones") is dynamically loaded when that user calls. A database front end, in which the application looks at the set of currently selected items, and dynamically builds a recognition grammar that allows the user to select from among those items. Any application in which the items to be recognized are different every time, and cannot be determined until runtime." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000918; see also id. at WF00000918-931 (Dynamic Grammars chapter)]. NSRS version 6 also included the ability to dynamically load a particular dictionary (or a set of word pronunciations) to be used in understanding a caller's query depending on context using the "WGISetPronunciation" function. [Id., at WF00000928].

KENT DECLARATION EXHIBIT 7

'977 Claim	Accused NSRS Features ¹	Prior Art NSRS Features
1. A speech-enabled internet website operating on a server computing system and compromising:	WF employs at least three separate IVR systems found to date at 800-642-4720 (banking); 800-368-7550 (Wells Fargo Advantage Funds) and 800-TRADERS (872-3377) (Wells Fargo Investments) that constitute part of a speech-enabled internet website. While testing has only been done extensively on the banking voice response system, the other systems are believed to perform the steps below as well. See e.g. PHO10434 – 10438 ("Speak your responses and the Automated Phone System will take care of the rest") and PHO10440 (tell the Automated Voice Response System what you would like to do") Wells Fargo infringes this claim in two separate ways. First, Wells Fargo operates an Internet website (www.wellsfargo.com) which together with the aforementioned IVRs constitutes a speech-enabled website. The WF IVR systems consist of hardware and software that recognizes speech based queries from WF customers so that the capabilities of WF's website (including web page items normally selectable by mouse clicks in a browser) are available through a separate speech interface.	A prior art NSRS had been implemented for Charles Schwab called the "VoiceBroker" application, which was announced as the "first telephone service using speech recognition technology to provide stock, mutual fund and market indicator information to retail customers." [Kent Decl., Ex. 9 (1996 article)] see also Sharp Decl., ¶ 28. Schwab also offered the same information on its website using a conventional keyboard/mouse through a browser. See [Kent Decl., Ex. 19 (Schwab Website)]. Further, the NSRS version 6 worked in conjunction with Nuance Voyager which "is a voice browser that significantly enhances the capabilities of the telephone through a voice interface that parallels that of a desktop Web browser. With Voyager, people can place phone calls and navigate between those calls and voice-enabled Web sites using spoken hyperlinks, or "voicelinks." Voyager is designed to allow phone users to tap into the wealth of information available on the Internet, all by using their voice and without hanging up the phone." [Kent Decl., Ex. 12 (Nuance Web)]. "Nuance Communications recently teamed with Motorola Inc. and Visa International to supplement Web-based commerce with speech-recognition capabilities. The technology will allow Web-based businesses to add integrated speech-recognition access to their services over ordinary phone lines. Three technologies were blended-Nuance speech objects, Motorola's Voice Markup Language (VoxML) and a Java application programmer interface-to allow users equal access to businesses from either the Web or the telephone.

¹ The accused NSRS features are quoted directly from Phoenix's Infringement Contentions and have been highlighted to make the comparison of key elements easier. Where Phoenix has accused multiple elements in the NSRS, the chart includes one or more of the elements that Phoenix contends are sufficient to prove infringement.

_

		This alliance enables users to conduct business over
		the Web, such as booking a plane reservation, and then access the same information over an ordinary telephone, such as changing that reservation while you are on the road," said Ronald Croen, president and chief executive officer of Nuance Communications.
		Nuance speech objects enable developers to assemble applications from a set of reusable speech-application components. Together with the Java API, developers can quickly build applications, in C++, Java or Visual Basic, that permit user access from either phone or Web. Motorola's VoxML simplifies the telephone access to Web-site business with a set of hypertext markup language (HTML) extensions that are used to integrate speech-recognition access to information collected from Web pages."
		[Kent Decl., Ex. 13 (TechSearch Article)]; see also [Kent Decl., Ex. 17 (Nuance Voyager Voice Browser)]; [Kent Decl., Ex. 18 (Nuance V-Builder)].
		See Sharp Decl., ¶ 33-34.
1.1 a receiving routine executing on the server computing system for receiving speech data	The WF IVRs include a speech recognition server, a type of computing system. The SR computing system is based on a speech recognition (SR) engine which recognizes words spoken continuously by a customer (a speech based query). [PHO003007; 2730 – 2731; 4143] This speech engine is embodied in software (Nuance 7.x/8.x) which must operate on a computing system referred to as a speech server. [PHO004155 - 4160; 3007] The hardware used by Wells Fargo is one of the systems described as sufficient for operating Nuance 7.x/8.x such as set out in PHO004160; PHO003007.	The NSRS version 6 included a speech recognition server referred to as the RecServer that received output speech data from the RecClient in order to perform speech recognition. This is shown in the below section: "Nuance6 capitalizes on the efficiency of processing of multiple recognition requests in parallel and provides automatic fail-over protection in the event of a hardware or software failure Nuance's client/server architecture also allows applications developers to decouple audio
	The speech recognition engine has a routine which receives speech data from a recognition client and/or a speech board.	applications developers to decouple audio acquisition from recognition processing. The RecClient performs all audio and telephony functions required by speech applications, including echo cancellation and endpointing when DSP is unavailable, and manages the collection and transmission of speech to the multi-threaded RecServer. The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user" [Kent Decl., Ex. 10 (Nuance6) at WF00000595].

The data flows from the RecClient to the

		RecServer.
		"Samples begin flowing to the recserver as soon as the onset of speech is detected, and the flow continues until the end of speech is found. In this way, the recserver can actually begin to process the utterance while the person is still speaking."
		[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000968; <i>see also id.</i> at WF00000726-727 (describing connection between RecServer and RecClient)].
		The RecServer receives the transmitted data and uses that data to recognize speech.
		"Consistent with the Nuance architecture, echo- canceled, endpointed audio data are recognized by the Nuance Recognition Server, running on the same host or another host."
		[Id. at WF00000903].
		"The processed speech values are then sent over a network connection to the RecServer which handles the speech recognition and understanding."
		[Kent Decl., Ex. 10 (Nuance6) at WF00000595].
		Thus, the recognition server had a routine that received speech data from the recognition client and/or the telephony board.
		See Sharp Decl., ¶ 16.
1.2 said speech data being characterized by a data content that is substantially inadequate by itself for	Such speech data must first be processed to extract spectral features, which are then input to the Hidden Markov Model (HMM) component of a speech recognizer. Hence the speech data as received is inadequate by itself for permitting recognition by the	As discussed above, in NSRS version 6, the speech data was transmitted from the recognition client to the recognition server for further processing, as shown in the below:
permitting recognition of words articulated in said speech query	HMM. [PHO3226-3229; 4118; 4241; 4280.]	"The RecClient performs all audio and telephony functions required by speech applications, including echo cancellation and endpointing when DSP is unavailable, and manages the collection and transmission of speech to the multi-threaded RecServer." [Kent Decl., Ex. 10 (Nuance6) at WF00000595].
		"The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user"
		[Kent Decl., Ex. 10 (Nuance6) at WF00000595].

		applied what is called Hidden Markov Models (HMMs) to recognize the words in the caller's speech as shown below: "The Nuance Speech Recognition System represents each word in the recognition vocabulary as a sequence of phonetic symbols. These symbols provide the correspondence between a word in the dictionary and its pronunciation." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000988]. "To recognize speech, the Nuance System compares it with a graph of acoustic speech models. In a
		process called the recognition search, the recognition engine searches this graph of possibilities for the sequence of models that best correspond to the speech. These models are found and interpreted as strings of words, and returned as the recognition result."
		[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000654].
		"Naturally, the Nuance System uses state-of-the-art HMMs at the core of its recognition engine.
		HMMs function primarily as acoustic models that provide a mapping from the sampled speech signal to a sequence of phonetic units."
		[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000964; <i>see also id.</i> WF00000965-968 (describing HMM)].
		See Sharp Decl., ¶ 17.
1.3 a speech recognition routine executing on the server computing	The Wells Fargo IVRs provide have a <i>speech</i> recognition (SR) engine that meets this limitation as acknowledged in WF materials (see PHO10434: "using advanced voice recognition") and the	In NSRS version 6, the speech recognition server (the RecServer) completes recognition of the speech query using HMM models:
system for completing recognition of said speech query using said speech data and	literature of the various vendors (including Intervoice, Nuance) who supplied the components of the IVR. (PHO010442-51; PHO010464 (Intervoice identified as source of Omvia speech enabled system);	"The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction
said data content to generate a recognized speech query	PHO010469; PHO010619-626 (Nuance further identified as assisting with system).	with the end user" [Kent Decl., Ex. 10 (Nuance6) at WF00000595].
	The SR routine (Nuance 7.x/8.x in this case) completes recognition of the speech query with an HMM using the speech data and the derived spectral features. [PHO3226-3229.] At this point	"To recognize speech, the Nuance System compares it with a graph of acoustic speech models. In a process called the recognition search, the recognition engine searches this graph of

the IVR system determines what the user wants.

The SR engine is further described at [PHO003007; 2730 – 2731; 4143] This speech engine is embodied in software (Nuance 7.x/8.x) [PHO004155 - 4160; 3007; PHO10511 (Omvia Speech Integration Module runs on a recognition server)] an hardware as set out in PHO004160; PHO003007; PHO010511-540].

Beyond this acknowledgement in WF's supplier literature, the existence of this SR engine behavior is also shown in the testing protocols provided by Phoenix on the WF system. [PHO10425 - 10433]

The fact that the system completely recognizes a speech query during is apparent from the audio clips also provided by Phoenix. [PHO10423, PHO10424]

possibilities for the sequence of models that best correspond to the speech. These models are found and interpreted as strings of words, and returned as the recognition result."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000654].

"Naturally, the Nuance System uses state-of-theart HMMs at the core of its recognition engine.

HMMs function primarily as acoustic models that provide a mapping from the sampled speech signal to a sequence of phonetic units."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000964; *see also id.* at WF00000965-968 (describing HMM)].

The NSRS version 6 works to recognize speech and returns a final recognition result:

"The RecResult structure contains the final recognition result, the NLResult, and several recognition statistics, including the confidence score of the recognition result. If N-best recognition was requested (by running the server with the parameter . . .), the RecResult structure may contain more than one recognition hypothesis. See the online documentation . . ., and associated functions for details on extracting recognition strings and other information from the RecResult structure."

[Id. at WF00000769].

See Sharp Decl., ¶ 19.

1.4 a web page having a list of items, at least some of said list of items being selectable by a user based on said recognized speech query Wells Fargo also maintains a web page – see www.wellsfargo.com

from which a number of items are selectable by a user using a conventional keyboard/mouse through a browswer. Some of these items are also selectable separately through the WF IVR system, as shown in audio clips. As an example, a user is allowed to transfer funds from their account through the website. The WF IVR allows a user to select this identical item (transfer funds) using a spoken query. (PHO010432)

To the extent that WF implements the speech based dialogs shown in PHO010425 – 433 using a VXML type architecture shown generally in PHO004580 (web pages accessible through a voice browser) such structure would also fall within this limitation. WF's vendors suggest that they have supplied such

NSRS had been implemented for Charles Schwab called the "VoiceBroker" application, which was announced as the "first telephone service using speech recognition technology to provide stock, mutual fund and market indicator information to retail customers."

[Kent Decl., Ex. 9 (1996 article)]; see also Sharp Decl., \P 28.

Schwab also offered the same information on its website using a conventional keyboard/mouse through a browser. Kent Decl., Ex. 19 (Schwab Website).

Further, NSRS version 6 worked in conjunction with Nuance Voyager which "is a voice browser that significantly enhances the capabilities of the

	functionality to WF: see e.g. PHO010619: (Nuance confirms that WF are using VoiceXML platforms in deployments)	telephone through a voice interface that parallels that of a desktop Web browser. With Voyager, people can place phone calls and navigate between those calls and voice-enabled Web sites using spoken hyperlinks, or "voicelinks." Voyager is designed to allow phone users to tap into the wealth of information available on the Internet, all by using their voice and without hanging up the phone." [Kent Decl., Ex. 12 (Nuance Web)]. "Nuance Communications recently teamed with Motorola Inc. and Visa International to supplement Web-based commerce with speech-recognition capabilities. The technology will allow Web-based businesses to add integrated speech-recognition access to their services over ordinary phone lines. Three technologies were blended-Nuance speech objects, Motorola's Voice Markup Language (VoxML) and a Java application programmer interface-to allow users equal access to businesses from either the Web or the telephone. This alliance enables users to conduct business over the Web, such as booking a plane reservation, and then access the same information over an ordinary telephone, such as changing that reservation while you are on the road," said Ronald Croen, president and chief executive officer of Nuance Communications.
		Nuance speech objects enable developers to assemble applications from a set of reusable speech-application components. Together with the Java API, developers can quickly build applications, in C++, Java or Visual Basic, that permit user access from either phone or Web. Motorola's VoxML simplifies the telephone access to Web-site business with a set of hypertext markup language (HTML) extensions that are used to integrate speech-recognition access to information collected from Web pages." [Kent Decl., Ex. 13 (TechSearch Article)]. See Sharp Decl., ¶ 33, 34.
1.5 wherein signal processing functions required to generate	rocessing functions a server computing system as noted above.	The NSRS version 6 operated on the Dialogic Antares telephony board.
said recognized speech query can be allocated between a client	4313-4332] The amount of processing used by either of these components can be allocated depending on resources available to each.	See [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000899-901].
platform and the		Such boards were capable of being configured to

server computing system as needed based on computing resources available to said client platform and server computing system respectively. This limitation is met by multiple facets of WF's IVR system(s), including one or more of the following:

- 1) As noted above the voice activity function can be selectively enabled depending on whether such resources are available on the particular type of speech board. [PHO004313 4317; 4249 4260; 2731 2737; 2739 -2742; 3007; 3915 3924; 4249 4260; 4313-4332; 4577 4578; 4490 4494]
- 2) Other speech processing operations performed by the recognition client, such as endpointing may also be selectively enabled and performed there or on the SR server. [PHO003140 3150]
- 3) For any WF IVR system using an SLM grammar, the degree of pruning of the SLM grammar by the server (second signal processing circuit) can be controlled [PHO003352]

handle more or less signal processing functions:

"Rather than offering a specific high level product, DSPSE offers the means to create custom products based on DSP technology with the Antares platform. DSPSE has 21 algorithm products that are compatible with the Antares platform and which can be used to quickly build high performance applications that fit specific customer requirements. The Systems Group at DSPSE has used these algorithm products to build "private label" solutions on the Antares platform. The combination of Antares specific experience and expertise in the development of DSP algorithms and applications enables DSPSE to quickly create the exact DSP based solution your application requires."

[Kent Decl., Ex. 15 (Antares Card)].

Further, in the NSRS version 6, the developer could selectively disable endpointing, and just specify the start and end of the speech:

"The RecClient API also allows the developer to specify the start and end of the utterance to be recognized, bypassing the spectral energy endpointing used by RCRecognize ().

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000770].

"When endpointing is not performed, all samples received by the client process are sent to the recognition server. Because users often pause after they are prompted to speak, some silence samples are unnecessarily sent to the recognition server. This increases the load put on the recognition server and lengthens the delay before the recognition server can respond."

[*Id.*, at WF00000968; *see also id.* at WF00000896-917 (Nuance Audio And Telephony Interfaces describing front end boards including Dialogic Antares card that handles echo-cancellation)].

A developer was also able to configure the endpointing parameters. [*Id.* at WF00000969].

Further, the NSRS version 6 permitted the configuration of the degree of pruning:

"The accuracy and speed of the Nuance System is determined by several factors: . . .

The amount of search performed. The recognition

See Sharp Decl., ¶ 19.	[Id. at WF00000877].	Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	"rec.PPR		This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail." [Id. at WF00000877].
the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases		between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658].		
Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases		between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks,		
"rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	"rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases	"rec.Pruning	between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks,		·
[Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	[Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases	[Id. at WF00000658]. "rec.Pruning			between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks,
This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases	This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning			[Id. at WF00000657].
"rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	"rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases	"rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning	[Id. at WF00000657].		speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be
speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases	speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning	speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased."		"rec.Pruning"
This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worses accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depend and the same and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks. [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	This parameter controls the trade-off between speed and accuracy by constraining the search space, By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks, and the properties of the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases	This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning	This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased."		[<i>Id.</i> at WF00000654-655].
This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during be search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digrecognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks. [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer perform sextra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depend on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases	This parameter controls the trade-off between speed and accuracy by constraining the search space, By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set ITRUE for IREQ-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning	This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased."		
optimized by varying system configurations. [Id. at WF00000654-655]. "rec.Pruning This parameter controls the trade-off between speed and accuracy by constraining the search space, By limiting the north of option of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypothese compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition task, the extra computations might cause longer response time. Therefore, the choice of TLE or FALSE for these parameters depends on the grammar and application at hand, it is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks, and FALSE for the speed/accuracy trade-off by varying the width of the Viterib beam search. A smaller number increases recognition speed, but decreases accuracy. See Chapter 2 for more detail."	optimized by varying system configurations. [Id. at WF00000654-655]. "rec.Pruning This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a directognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning Controls the speed/accuracy trade-off by varying the width of the Viterbi beam search. A smaller number increases recognition speed, but decreases	optimized by varying system configurations. [Id. at WF00000654-655]. "rec.Pruning This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased." [Id. at WF00000657]. "rec.PPR This Boolean parameter controls the trade-off between speed and accuracy by looking ahead to prune less likely hypotheses from further consideration during the search. In doing so, the recognizer performs extra computations using approximate models. If the grammar is large, the pruning of unlikely hypotheses compensates for the extra computations needed and, hence, the net effect is a faster response time with slightly worse accuracy. On the other hand, if the search space is small, such as in a digit recognition or a small-vocabulary command recognition task, the extra computations might cause longer response time. Therefore, the choice of TRUE or FALSE for these parameters depends on the grammar and application at hand. It is recommended that this parameter always be set to TRUE for large-vocabulary tasks, and FALSE for small-vocabulary tasks." [Id. at WF00000658]. "rec.Pruning	optimized by varying system configurations. [Id. at WF00000654-655]. "rec.Pruning This parameter controls the trade-off between speed and accuracy by constraining the search space. By limiting the number of different search possibilities carried along, recognition speed can be increased."		This will speed up the system but might cause

6. The website of claim 1, wherein said website is further adapted to respond to a speech query concerning said list of items by returning a text or speech articulated response.

WF's IVR returns speech articulated responses as seen and observed in the protocols and audio scripts. [PHO10425 - 10433] and the audio files [PHO10423 - 10424]

IVR systems implementing NSRS returned speech articulated responses.

This was true in the Schwab Voice Broker implementation:

See, e.g., [Kent Decl., Ex. 9 (1996 VoiceBroker Press Release) at WF00000593-594].

It was also true as to NSRS version 6 in the Nuance demonstrations.

[Kent Decl., Ex. 10 (Nuance6) at WF00000597-604].

Further, the NSRS version 6 worked with Entropic's TrueTalk text-to-speech system:

"The only text-to-speech system that Nuance has an interface to is Entropic's TrueTalk. However, you may be able to call other systems directly from your application code. Contact Nuance technical support for more information.

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000760].

The NSRS version 6 was also capable of playing audio files:

"The RecClient API provides two functions that play audio out through the audio device: RCPlayFile () and RCPlayLastUtterance ().

RCPlayFile () and RCPlayLastUtterance (). RCPlayFi le () takes as an argument a character string listing one or more audio files, separated by spaces, which should be played out the audio device. The files are concatenated and played as a unit."

[Id. at WF00000772].

"The Nuance System includes a waveform editor, Xwavedit, which makes it easy to record, play, view, crop, cut, and paste waveforms. The Xwavedit program is ideal for recording prompts and removing trailing silence from them. For full details on Xwavedit, see the online documentation.

[Id. at WF00000958].

"Nuance provides several programs you can use to normalize .wav files. Nuance recommends that you normalize audio files used as prompts.

wavnorm Lets you normalize a single .wav file,

		specifying the RMS. The recommended value is 2200. wavdynanomz Lets you normalize a set of .wav files. wavrms Lets you determine the average energy level of speech in a set of .wav files. wavxform Lets you perform operations such as clipping and filtering on .wav files." [Id. at WF00000960].
		See Sharp Decl., ¶ 29.
7. The website of claim 1, wherein said website is further adapted to interact on a real-time basis in response to one or more continuous speech queries.	The WF IVR responds in real-time to one or more continuous speech queries as seen and observed in the protocols and audio scripts. [PHO10425 - 10433] and the audio files [PHO10423 - 10424]	The NSRS version 6 operated in real-time. Indeed, "[s]amples begin flowing to the recserver as soon as the onset of speech is detected, and the flow continues until the end of speech is found. In this way, the recserver can actually begin to process the utterance while the person is still speaking." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000968]. See Sharp Decl., ¶ 15.
10. The website of claim 1, wherein the website also controls an interactive character agent presented to the user for assisting in handling said speech query.	WF's interactive agent is monitored and observed in the test protocols and audio scripts. [PHO10425 - 10433] and the audio files [PHO10423 - 10424]	The NSRS version 6 included the capability to monitor the system performance, including audio scripts. [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000663] ("One way to check the speech signal is to use Xwavedit to examine recordings made by your application. You should listen and look for endpointing errors, signal distortions (e.g., clipping), very low amplitude signals, and recordings that just sound bad."). See Sharp Decl., ¶ 32.

KENT DECLARATION EXHIBIT 8

6854 Claim	A coursed NSDC Factures	Duion Aut NCDC Footuuga
'854 Claim	Accused NSRS Features ¹	Prior Art NSRS Features
1. A method of interacting with	The Wells Fargo interactive voice response (IVR) systems include a natural language query	The NSRS version 6 enables a method of interacting with
a user of a	system (hardware and software) that interacts	customers who speak naturally to the system:
natural language query system comprising the steps of:	with Wells Fargo customers (users) using the steps of the method described below. WF employs at least three separate IVR systems found to date at 800-642-4720 (banking); 800-368-7550 (Wells Fargo Advantage Funds) and 800-TRADERS (872-3377) (Wells Fargo Investments). While testing has only been done extensively on the banking voice response system, the other systems are believed to perform the steps below as well. See e.g. PHO10434 – 10438 ("Speak your responses and the Automated Phone System will take care of the rest") and PHO10440 (tell the Automated Voice Response System what you would like to do").	"Nuance6, the core client/server software in the Nuance Communications Conversational Transactions product suite, combines unparalleled accuracy and natural language understanding to extend the performance advantage of Nuance in speech recognition applications. Utilizing advanced linguistic and statistical models to interpret and understand natural human speech, Nuance6 enables sophisticated speech recognition applications which allow users to talk to computers as if they were speaking with human agents. Nuance6's open, scalable software architecture, comprised of the Nuance RecServer and the Nuance RecClient, makes efficient use of hardware resources to allow the implementation of affordable, flexible speech recognition applications that give users a competitive advantage in today's rapidly changing business environment. [Kent Decl., Ex. 10 (Nuance6) at WF00000595].
		Sharp Decl., ¶ 8.
1.1 a) providing a speech recognition engine adapted to recognize a first set of words and/or phrases from a	The Wells Fargo IVRs provide have a <i>speech</i> recognition (SR) engine that meets this limitation as acknowledged in WF materials (see PHO10434: "using advanced voice recognition") and the literature of the various vendors (including Intervoice, Nuance) who supplied the components of the IVR. (PHO010442-51; PHO010464 (Intervoice	NSRS version 6 recognizes the natural, continuous speech of a customer during an interactive speech session: "Utilizing advanced linguistic and statistical models to interpret and understand natural human speech, Nuance 6 enables sophisticated speech recognition applications that allow users to talk to computers as if they were speaking with human agents."
user during an interactive speech session;	identified as source of Omvia speech enabled system); PHO010469; PHO010619-626 (Nuance further identified as assisting with system).	[Kent Decl., Ex. 14 (Nuance6 Data Sheet)].
wherein said first set of words and/or phrases can include a natural	The SR engine recognizes the set of words/phrases spoken continuously by a customer for the query during an interactive speech session. [PHO003007; 2730 – 2731; 4143] This speech engine is embodied in software	"The Nuance speech recognition engine is both powerful and flexible. It provides speaker-independent speech recognition capability, can recognize from lists of 15,000 or more phrases, and can extract meaning from naturally spoken sentences."
language query presented as continuous	(Nuance 7.x/8.x) which must operate on a speech server computer. [PHO004155 - 4160; 3007; PHO10511 (Omvia Speech Integration Module	[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000634];
natural language spoken data;	runs on a recognition server)] The hardware used by Wells Fargo for implementing this is described for operating Nuance 7.x/8.x such as set out in PHO004160; PHO003007; PHO010511 -540].	"To recognize speech, the Nuance System compares it with a graph of acoustic speech models. In a process called the recognition search, the recognition engine searches this graph of possibilities for the sequence of models that best

¹ The accused NSRS features are quoted directly from Phoenix's Infringement Contentions and have been highlighted to make the comparison of key elements easier. Where Phoenix has accused multiple elements in the NSRS, the chart includes one or more of the elements that Phoenix contends are sufficient to prove infringement.

1

Beyond this acknowledgement in WF's supplier literature, the existence of this SR engine behavior is also shown in the testing protocols provided by Phoenix on the WF system. [PHO10425 - 10433] The fact that the system recognizes words during an interactive session spoken as a natural language query is apparent from the audio clips provided by Phoenix. [PHO10423 - 424]

The vocabulary of words which the system is programmed to understand are embodied in a structure referred to as a SR "grammar." The Wells Fargo IVRs use grammars trained to respond to Wells Fargo-specific dialog data, such as Wells Fargo product/service offerings. [PHO010442 - 480; 10619- 10626; 4462 - 4463]

correspond to the speech. These models are found and interpreted as strings of words, and returned as the recognition result."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000654; *see also id.* at WF00000964-968 (describing HMM)].

"The RecResult structure contains the final recognition result, the NLResult, and several recognition statistics, including the confidence score of the recognition result. If N-best recognition was requested (by running the server with the parameter . . .), the RecResult structure may contain more than one recognition hypothesis. See the online documentation . . . , and associated functions for details on extracting recognition strings and other information from the RecResult structure."

[Id. at WF00000769].

See Sharp Decl., ¶ 8.

1.2 b) providing a database of query/answer pairs concerning one or more topics which can be responded to by the natural language query system during said interactive speech based session with a user;

The WF IVRs provide at least one electronic database that includes pairs of questions and associated answers on topic it can cover with customers. For example, as seen in [PHO10425 - 10433] some of the topics covered by the IVR at 800-642-4720 is a customer's balance or an account history. Thus in response to the query for "balance" the system gives an answer that indicates the customer's account balance.

One of the topics covered by the Banking demonstration of NSRS version 6 was a customer's balance or an account history. Thus in response to the query for "balance" the system gives an answer that indicates the customer's account balance.

See [Kent Decl., Ex. 10 (Nuance6) at WF00000599-600 (describing Banking Demonstration)].

Further information is provided in the Nuance Developer's Manual version 6, [Sharp Decl. Ex. A at WF00000686-687, WF00000739, WF00000956, WF00001012-1022].

Indeed, on page 317, the Manual provides "a sample banking application built with the Dialog Builder" that "lets you check your balance, transfer money, pay bills, and add new payees."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000956].

See Sharp Decl., ¶ 26

1.3 c) providing a natural language routine adapted to process said first set of words and/or phrases and identify a response to said The Wells Fargo IVRs provide a *natural language* (NL) routine that meets this limitation as acknowledged in the literature of the various vendors who supplied the components of the IVR. See e.g. PHO002806 "...The Nuance System lets you define natural language understanding in your recognition packages. A natural language understanding system takes a sentence (typically a recognized utterance) as input and returns an interpretation—a representation of

NSRS version 6 also had the option of including what phoenix refers to a "natural language routine" that took a sentence (typically a recognized utterance) as input and returns an interpretation—a representation of the meaning of the sentence:

"The Nuance System allows developers to quickly and easily create powerful natural language understanding systems tailored to their needs. A natural language understanding system takes a sentence as input and

natural language query based on said query/answer pairs; the meaning of the sentence." See also PHO0002807-2821.

This routine is used to determine the meaning of the user's query based on the words/phrases identified by the SR engine. This NL engine is also embodied in software, and typically operates on the speech server as well. In the case of the WF IVR this engine is embodied either as part of the Nuance 7.x/8.x software

returns as output an interpretation, which is a representation of the meaning of the sentence. . . .

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000678].

"Once the RecServer has recognized the speech, it uses a template-matching natural language approach to determine meaning. This means the RecServer takes the relevant pieces of recognized speech and matches them against a customized template of pre-specified information. This template approach is more robust and sophisticated than word spotting or "tag" approaches and easier and more efficient than an application-specific sentence parser."

[Kent Decl., Ex. 10 (Nuance6) at WF00000596].

"Natural language understanding applications built with the Nuance System produce interpretations consisting of slots and values for these slots. For any given application, the developer specifies a fixed set of slots, which correspond to types of information commonly supplied in queries in the application's domain. For example, in an automated banking application, slots might include command-type, amount, source-account, and destination account.

The interpretation for the command "Transfer five hundred dollars from savings to checking" would then probably fill the command-type slot with transfer, the source-account slot with savings, the destination-account slot with checking, and the amount slot with 500."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000678; *see also id.* at WF00000679-686].

See Sharp Decl., ¶ 26.

1.4 wherein said natural language routine is adapted to consider only a subset of said first set of words and/or phrases, and further can consider words and/or phrases in said natural language query which are not present in said query/answer

The WF NL routine behavior is also shown in the testing protocols provided by Phoenix. [PHO10425 - 10433] In these routines it can be seen that the NL routine only considers some of the words/phrases (such as the phrase "account history") in the query. Moreover the NL routine can consider other words/phrases (such as "ah" "uhmm" and other dysfluencies, as well as extra words like "give me my") presented by the SR engine output that are not even present in the questions/answers and still determine the response to the query. See e.g. PHO PHO010642 ("...With Nuance 8.0, companies can build NVP applications that recognize a variety of speech inputs—from simple "yes/ no" responses to natural sentences like "hi... I'd like to check my account balance.") and Say Anything technologies explained at

The NSRS version 6 allowed the recognition and understanding of words/phrases (such as the phrase "account history") in the query and could determine the appropriate response to a query even in the presence of other words/phrases such as dysfluencies and extra words.

First, the NSRS version 6 was able to understand phrases:

"Transfer five hundred dollars from savings to checking" would then probably fill the command-type slot with transfer, the source-account slot with savings, the destination-account slot with checking, and the amount slot with 500."

[Id. at WF00000678].

Second, the NSRS version 6 was able to return a result even in the presence of dysfluencies or extra words because the

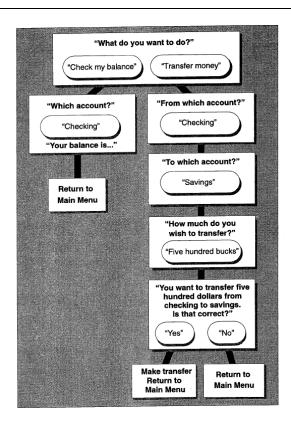
pairs to	PHO003338 – 3361.	recognition server would not recognize extraneous words
determine said response;	The subset of words/phrases which the NL routine looks for are embodied part of a natural language grammar which Wells Fargo again has customized for its particular business/customer operations. [PHO003320 - 3361; 4473] See also the literature above for the various functions offered by the three separate IVRs used by WF.	[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000661-662 (describing need to include compound words in grammar so that they could be recognized)]. Third, as a template matching system, the "natural language routine" only considered the meaning of the words/phrases that corresponded to slots. "Natural language understanding applications built with the Nuance System produce interpretations consisting of slots and values for these slots. For any given application, the developer specifies a fixed set of slots, which correspond to types of information commonly supplied in queries in the application's domain. For example, in an automated banking application, slots might include command-type, amount,
		source-account, and destination account." [Id. at WF00000678; see also id. at WF00000679-686; WF00001012-1022 (providing examples)]; see also [Kent Decl., Ex. 10 (Nuance6) at WF00000599-600 (describing Banking Demonstration)]. Fourth, this was shown in the Nuance demonstrations.
		[Kent Decl., Ex. 10 (Nuance6) at WF00000596].
		See Sharp Decl., ¶ 20
1.5 d) providing an interactive electronic agent coupled to said natural language routine and	The Wells Fargo IVRs provide an interactive electronic agent (as acknowledged in WF's own press releases above) which provides the answer to the customer after it is determined by the SR/NL routines. This electronic agent is presented to customers in the form of an artificial persona that articulates answers in audible form.	The NSRS version 6 was intended to be used, and was in fact used, in IVR systems where there was an agent that articulates answers (<i>i.e.</i> , the response to the user) in audible form. See, e.g., [Kent Decl., Ex. 10 (Nuance6) at WF00000597-604 (describing Demonstrations)].
configured to:	As seen in testing protocols provided by Phoenix [PHO10425 - 10433] and the audio files [PHO10423 - 10424] the electronic agent	The audible speech was capable of providing a prompt regarding possible user queries, confirming the substance of the query, and providing an answer or does the requested
i. provide a prompt to the user during said interactive speech based session with	provides specific suggestions to callers on what types of queries can be made ("We're back at the main menu, you can say balance, account history, transfer funds or other services") [PHO10432]	action. For example, the following sample dialogue suggested that the caller should suggest an amount to transfer, confirmed the amount specified by the caller, and then made the transfer:
suggestions on queries which can be made to the natural language query	As above, note that the agent repeats and confirms the query made by the caller "JNG: I'd like you to repeat my balance please. WFB: All right, repeat balance. [PHO10432]	

The agent then goes on to provide the account

system;

ii. provide a confirmation of a substance of said natural language query;

iii. provide said response to the user from the natural language query routine." **balance**: "As of February 14th..." (the rest is removed for privacy considerations).



[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000739].

This is further described below:

"To build an application with the Dialog Builder, you create a set of dialog states. A dialog state provides code that handles a single interchange between the user and the computer. For example, a banking application might have a state in which the application asks the user "How much do you want to transfer?" and the user responds with a dollar amount that the system would recognize. After a successful interchange, an application typically moves into a new dialog state-for example, the banking application might move into a confirmation state after the transfer amount state. You can envision the collection of interconnected dialog states in an application as a network."

[Id. at WF00000738].

This is further described below:

"Here is an example of a state function for a state that gets a money amount from the user (lines are numbered for reference): . . .

Line-by-line explanation follows:

		8: how-much .wav is a file that contains a recording of the prompt "How much do you want to pay?" 9: AppRecognize () plays the appended prompts, starts recognition, and waits for recognition to finish 16-18: The system prompts something like "You specified fifty dollars".
7. The method of claim 1 further including a step: configuring perception related parameters of the electronic interactive agent based on preferences of said user, including one of a gender, a visual appearance, and/or voice characteristics including one of pitch, volume and/or speed.	WF's electronic agents are configured with parameters (such as the choice of a female gender for the voice) based on the WF's understanding of the preferences of its customers. See e.g. PHO010444 "provide a business-like, efficient, professional voice that's task and audience appropriate" See also PHO010445 – 47; PHO010452 – 454;	The NSRS version 6 system permits for recording of prompts that can be configured based on gender, or voice characteristics such as pitch, volume and/or speed. "The RecClient API provides two functions that play audio out through the audio device: RCPlayFile () and RCPlayLastUtterance (). RCPlayFile () takes as an argument a character string listing one or more audio files, separated by spaces, which should be played out the audio device. The files are concatenated and played as a unit." [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000772]. "The Nuance System includes a waveform editor, Xwavedit, which makes it easy to record, play, view, crop, cut, and paste waveforms. The Xwavedit program is ideal for recording prompts and removing trailing silence from them. For full details on Xwavedit, see the online documentation." [Id. at WF00000958]. "Nuance provides several programs you can use to normalize wav files. Nuance recommends that you normalize audio files used as prompts. wavnorm Lets you normalize a single wav file, specifying the RMS. The recommended value is 2200. wavdynanomz Lets you normalize a set of .wav files. wavrms Lets you determine the average energy level of speech in a set of .wav files.

		wavxform Lets you perform operations such as clipping and filtering on .wav files." [Id. at WF00000960]. "For recording prompts, you may want to use Xwavedit. You can crop each prompt with Xwauedit, and then normalize your prompts with wavnom." [Id. at WF00000750]. Thus, NSRS permitted the recording of prompts that were based on the designer's understanding of its customers' preference including one of gender—e.g., a developer could have recorded a female voice for the prompt, if that developer believed that was his or her customer's preference.
		See Sharp Decl., ¶ 31.
8. The method of claim 1 wherein said subset of words and/or phrases in said natural language query can be assigned different weightings determined by said natural language routine.	It is apparent from the choices made by the WF IVR system that it assigns different weights to certain words as it favors some words and ignores others completely.	NSRS also was capable of favoring some words, for example, by including the words in the grammar (which permitted them to be recognized). [Sharp Decl., Ex. A (Nuance Developer's Manual at WF00000654-655)]. Further, the NSRS version 6 system assigned different weights to certain words by recognizing them as corresponding to predefined slots such as "command-type, amount, source-account, and destination account": "Natural language understanding applications built with the Nuance System produce interpretations consisting of slots and values for these slots. For any given application, the developer specifies a fixed set of slots, which correspond to types of information commonly supplied in queries in the application's domain. For example, in an automated banking application, slots might include command-type, amount, source-account, and destination account."). [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000678]; see also [Kent Decl., Ex. 10 (Nuance6) at WF00000596] ("This means the RecServer takes the relevant pieces of recognized speech and matches them against a customized template of pre-specified information."). Weights also could be assigned to the acoustic versus linguistic scores during recognition: "This parameter controls the grammar processing weight-the relative weighting of acoustic and linguistic scores during

		recognition. This is meaningful only when probabilities in your grammar are used. (See Chapter 1 for more detail about grammar probabilities.) The relative weight of the grammar probabilities versus the acoustic scores is controlled by rec. GrammarWeight. A large value for this parameter emphasizes the linguistic constraints set by the probabilities, and a small value emphasizes the acoustic scores. The rec. GrammarWeight interacts directly with rec. Pruning. [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000658]. "rec.GrammarWeight Controls the grammar processing weight-the relative weighting of acoustic and linguistic scores during recognition. See Chapter 2 for more detail." [Id. at WF00000878]. Sharp Decl., ¶ 21, 22.
9. The method of claim 1, wherein said speech recognition engine is distributed between a client device and a server system which receives streaming speech data having reduced latency data content before silence is detected and the utterance is complete.	In Wells Fargo's IVR system, the speech recognition functions are distributed between a client device and a server device. For example certain functions such as end pointing and voice activity detection (VAD) are performed by the SR engine component at the client, while the remainder are done on the recognition server. PHO004156; PHO010513; PHO010530; The client portion of the SR engine is configured to send data only while the user is talking (before silence and the utterance being complete) thus reducing the amount of data otherwise associated with the user utterance. [PHO004313 – 4317; 4249 – 4260; 2739 -2742; 4318-4332; 4577 – 4578; 4490 – 4494]	NSRS version 6 included a recognition client that processed the input speech signals by (among other things) performing echo cancellation and endpointing: "Nuance6 capitalizes on the efficiency of processing of multiple recognition requests in parallel and provides automatic fail-over protection in the event of a hardware or software failure Nuance's client/server architecture also allows applications developers to decouple audio acquisition from recognition processing. The RecClient performs all audio and telephony functions required by speech applications, including echo cancellation and endpointing when DSP is unavailable, and manages the collection and transmission of speech to the multi-threaded RecServer. The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user" [Kent Decl., Ex. 10 (Nuance6) at WF00000595]. "The RecClient performs audio and telephony functions, freeing applications to focus on higher-level issues such as call flow and dialog management The RecClient then attaches to the audio device for the application-for example, a telephone port-and provides the application with the following functionality: - Energy-based endpointing to determine the beginning and end of the talker's speech

- Recognizing speech via dynamic connections to Nuance
recognition servers."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000717].

"Because a speech recognition application actually performs recognition only a certain percentage of the time, the RecServer was designed to be used intermittently by multiple RecClients running different applications. This allows the CPU and memory resources of the RecServer to be shared across multiple client applications, and provides the following advantages:

- Distributed Architecture. Recognition, which can be CPU-intensive, can be run on a different machine from the machine running the application and audio interface.

[Id. at WF00000724].

NSRS version 6 also included the capability of using a Dialogic telephony board that could process the input speech signals by (among other things) performing echo cancellation and endpointing:

"The Nuance System RecClient supports several telephone interfaces, including Dialogic."

[Id. at WF00000774].

"Consistent with the Nuance architecture, echo-canceled, endpointed audio data are recognized by the Nuance Recognition Server, running on the same host or another host."

[*Id.* at WF00000903; *see also id.* at WF00000896, WF00000899-904 (describing Nuance Audio And Telephony Interfaces including Dialogic Antares card that handles endpointing and echo-cancellation)].

The data output from the RecClient was then passed to the RecServer which generated recognized words:

"Samples begin flowing to the recserver as soon as the onset of speech is detected,"

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000968].

"To recognize speech, the Nuance System compares it with a graph of acoustic speech models. In a process called the recognition search, the recognition engine searches this graph of possibilities for the sequence of models that best correspond to the speech. These models are found and interpreted as strings of words, and returned as the

recognition result."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000654; *see also id.*. at WF00000964-968 (describing HMM)].

"The RecResult structure contains the final recognition result, the NLResult, and several recognition statistics, including the confidence score of the recognition result. If N-best recognition was requested (by running the server with the parameter . . .), the RecResult structure may contain more than one recognition hypothesis. See the online documentation . . . , and associated functions for details on extracting recognition strings and other information from the RecResult structure."

[Id. at WF00000769].

The client portion of the SR engine is configured to send only a caller's speech (thus reducing the amount of data otherwise associated with the user utterance), and to do so, before silence and the utterance being complete:

Endpointing "allows more computationally efficient use of a recognition server's resources. When endpointing is not performed, all samples received by the client process are sent to the recognition server."

[Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000968],

Endpointing is used to prevent the following:

"Because users often pause after they are prompted to speak, some silence samples are unnecessarily sent to the recognition server. This increases the load put on the recognition server and lengthens the delay before the recognition server can respond."

Id.

Thus, when endpointing is used, only the speech that is detected is transmitted to the SR server:

Samples begin flowing to the recserver as soon as the onset of speech is detected, and the flow continues until the end of speech is found. In this way, the recserver can actually begin to process the utterance while the person is still speaking.

Id.

This is also reflected by fact that "[s]ometimes the word that the user says may not have enough energy in the beginning of the word to set off the endpointer. As a result, only the middle portion of the word is sent to the recognizer."

		[<i>Id.</i> at WF00000969].
		Sharp Decl., ¶ 15.
13. The method of claim 1 wherein the interactive electronic agent provides responses adjusted for a context experienced by the user.	The grammar used by the IVR system depends on where the user is in the dialog state. For example, the main menu grammar (see example of choices at PHO010427) is different from the top level grammar which just receives the account number (id.) See also: "The IVR presents a recognition context along with a speech utterance to the recognition server selected by the resource manager. This constrains the recognition task to one of a finite number of preloaded recognition tasks on the server." PHO PHO010514 That the Wells Fargo systems use grammars specific to context is also acknowledged by reference to the vendor literature showing that this is a basic feature of Nuance 7.x/8.x [PHO004158; PHO002998; PHO003155; PHO003362]	NSRS version 6 allowed where a caller was in the dialogue to play a role in recognizing the caller's speech—e.g., the recognition server knew if the caller had just told the system that she wanted to transfer money and thus interpreted the caller's next speech in light of that knowledge. This is accomplished using the function "AppSetGrammar." See [Sharp Decl., Ex. A (Nuance Developer's Manual at WF00000751)]. Further, the NSRS version 6 would use "contextualization" so that, for example, when a user said "transfer today," the recognition server would understand that the word "today" referred to a particular date based on the context of the utterance: "Contextualization refers to the process of updating something to reflect the context of utterance. The function ContextualizeDate () updates a date value to reflect the time at which it was uttered." [Id. at WF00000758]. Indeed, "[t]he usability of Nuance applications is enhanced by advanced features, such as single phrase correction, that ensures users do not need to repeat an entire sentence if only part of what they said is not recognized." [Kent Decl., Ex. 16 (Nuance6 1999 Data Sheet)]. This is also shown in the review of NSRS version 6, where the banking demonstration which could understand where the user was in the dialogue. [Kent Decl., Ex. 11 (Oct. 1998 Review)].
15. An electronic agent based natural language query system comprising:	The Wells Fargo interactive voice response (IVR) systems include a natural language query system (hardware and software) that interacts with Wells Fargo customers (users) using the steps of the method described below. WF employs at least three separate IVR systems found to date at 800-642-4720 (banking); 800-368-7550 (Wells Fargo	See limitation 1.

	Advantage Funds) and 800-TRADERS (872-3377) (Wells Fargo Investments). While testing has only been done extensively on the banking voice response system, the other systems are believed to perform the steps below as well. See e.g. PHO10434 – 10438 ("Speak your responses and the Automated Phone System will take care of the rest") and PHO10440 (tell the Automated Voice Response System what you would like to do") Each of the IVR systems uses an automated speaking persona which imitates a human to interact with Wells Fargo customers.	
15.1 a) a database of query/answer pairs concerning one or more topics which can be responded to by the natural language query system during an interactive speech based session with a user;	The WF IVRs include at least one electronic database that includes pairs of questions and associated answers on topic it can cover with customers. For example, as seen in [PHO10425 - 10433] some of the topics covered by the IVR at 800-642-4720 is a customer's balance or an account history. Thus in response to the query for "balance" the system gives an answer that indicates the customer's account balance. Sometimes the answer provided by IVR, as soon as she determines the customer's query, is as simple as finding a person who can give the rest of the answer on the topic. See e.g. PHO10427.	See limitation 1.2.
15.2 b) a speech recognition engine executing at a computing system and adapted to recognize a first set of words and/or phrases from a user during said interactive speech session; wherein said first set of words and/or phrases can include a natural language query presented as continuous	The Wells Fargo IVRs include a <i>speech recognition</i> (<i>SR</i>) <i>engine</i> that meets this limitation as acknowledged in WF materials (see PHO10434: "using advanced voice recognition") and the literature of the various vendors (including Intervoice, Nuance) who supplied the components of the IVR. (PHO010442-51; PHO010464 (Intervoice identified as source of Omvia speech enabled system); PHO010469; PHO010619-626 (Nuance further identified as assisting with system). The SR engine recognizes the set of words/phrases spoken continuously by a customer for the query during an interactive speech session. [PHO003007; 2730 – 2731; 4143] This speech engine is embodied in software (Nuance 7.x/8.x) which must operate on a speech server computer. [PHO004155 - 4160; 3007; PHO10511 (Omvia Speech Integration Module runs on a recognition server)] The hardware used by Wells Fargo for implementing this is described for operating Nuance 7.x/8.x such as set out in PHO004160; PHO003007; PHO010511 -540].	See limitation 1.1.

		,
natural language spoken data;	Beyond this acknowledgement in WF's supplier literature, the existence of this SR engine behavior is also shown in the testing protocols provided by Phoenix on the WF system. [PHO10425 - 10433] The fact that the system recognizes words during an interactive session spoken as a natural language query is apparent from the audio clips provided by Phoenix. [PHO10423 - 424] The vocabulary of words which the system is programmed to understand are embodied in a structure referred to as a SR "grammar." The Wells Fargo IVRs use grammars trained to respond to Wells Fargo-specific dialog data, such as Wells Fargo product/service offerings. [PHO010442 - 480; 10619- 10626; 4462 - 4463]	
15.3 c) a natural language routine executing at said computing system and adapted to process said words and/or phrases and identify a response to said natural language query derived from said query/answer pairs;	The Wells Fargo IVRs provide a <i>natural language</i> (NL) routine that meets this limitation as acknowledged in the literature of the various vendors who supplied the components of the IVR. See e.g. PHO002806 "The Nuance System lets you define natural language understanding in your recognition packages. A natural language understanding system takes a sentence (typically a recognized utterance) as input and returns an interpretation—a representation of the meaning of the sentence." See also PHO0002807-2821. This routine is used to determine the meaning of the user's query based on the words/phrases identified by the SR engine. This NL engine is also embodied in software, and typically operates on the speech server as well. In the case of the WF IVR this engine is embodied either as part of the Nuance 7.x/8.x software and/or other natural language software provided by Intervoice. [PHO003320 - 3361; 4473] This routine may also be present in any products used by Wells Fargo for directing calls. <i>See e.g.</i> PHO010632 – 639 "callers interact with an automated speech recognition-based solution that enables them to speak naturally and then to be quickly routed to the correct destination."	See limitation 1.3.
15.4 wherein said natural language routine is adapted to consider only a subset of said words and/or	The WF NL routine behavior is also shown in the testing protocols provided by Phoenix. [PHO10425 - 10433] In these routines it can be seen that the NL routine only considers some of the words/phrases (such as the phrase "account history") in the query. Moreover the NL routine can consider other words/phrases (such as "ah" "uhmm" and other dysfluencies, as well as extra	See limitation 1.4.

phrases, and can consider words and/or phrases in said natural language query which are not present in said query/answer pairs to determine said response;	words like "give me my") presented by the SR engine output that are not even present in the questions/answers and still determine the response to the query. See e.g. PHO PHO010642 ("With Nuance 8.0, companies can build NVP applications that recognize a variety of speech inputs—from simple "yes/ no" responses to natural sentences like "hi I'd like to check my account balance.") and Say Anything technologies explained at PHO003338 – 3361.	
15.5 d) an interactive agent routine coupled to said natural language routine and for providing an interactive electronic agent configured and controlled by one or more additional second routines to: i. provide a prompt to the user during said interactive speech based session with suggestions on queries which can be made to the natural language query system; ii. provide a confirmation of a substance of said natural language query; iii. provide said response to the user from the natural	The Wells Fargo IVRs provide an interactive electronic routine that meets this limitation as acknowledged in WF's own press releases above; the interactive electronic agent provides the answer to the customer after such answer is determined by the SR/NL routines. This electronic agent is presented to customers in the form of an artificial persona that articulates answers in audible form. As seen in testing protocols provided by Phoenix [PHO10425 - 10433] and the audio files [PHO10423 - 10424] the electronic agent provides specific suggestions to callers on what types of queries can be made ("We're back at the main menu, you can say balance, account history, transfer funds or other services") [PHO10432] As above, note that the agent repeats and confirms the query made by the caller "JNG: I'd like you to repeat my balance please. WFB: All right, repeat balance. [PHO10432] The agent then goes on to provide the account balance: "As of February 14th" (the rest is removed for privacy considerations).	See limitation 1.5.

language routine.		
19. The system of claim 15 further including a routine to configure perception related parameters of the electronic interactive agent, including one of a gender, a visual appearance, and/or voice characteristics including one of pitch, volume and/or speed.	WF's electronic agents are configured by a routine with parameters (such as the choice of a female gender for the voice) based on the WF's understanding of the preferences of its customers. See e.g. PHO010444 "provide a business-like, efficient, professional voice that's task and audience appropriate" See also PHO010445 – 47; PHO010452 – 454;	See limitation 7.
20. The system of claim 15, wherein at least one of said prompt, confirmation or response is processed by a text to speech engine and rendered into audible form for the user by the interactive electronic agent.	WF uses a conventional text to speech engine to render the response into audible form for the user. See e.g. PHO004008 - Omvia Speech Integration Module—RealSpeak text-to-speech & Speechify text-to-speech; PHO003504 (The Nuance Conversation Server includes a VoiceXML Interpreter integrated with Nuance's industry leading speech recognition, text-to-speech and voice authentication technologies.) See also PHO010444 - 47	See limitation 13. Further, NSRS version was capable of being operated with Entropic's TrueTalk text-to-speech system: "The only text-to-speech system that Nuance has an interface to is Entropic's TrueTalk. However, you may be able to call other systems directly from your application code. Contact Nuance technical support for more information. [Sharp Decl., Ex. A (Nuance Developer's Manual version 6) at WF00000760]. See Sharp Decl., ¶ 29.
22. The system of claim 15 wherein said subset of words and/or phrases in said natural language query	It is apparent from the choices made by the WF IVR system that it assigns different weights to certain words as it favors some words and ignores others completely. This weighting can be derived manually or from	See limitation 8.

can be assigned different weightings determined by said natural language routine.	statistical analysis of call data. See e.g. PHO PHO003361 " Through statistical methods, this tool discovers the correlations between routing destinations and key phrases." The fact that the IVR system uses weightings is also believed to be confirmed by the vendor's US Patent No. 6,856,957, col. 4, ll. 40 - 41 which explains how a query "vector" with weightings is formed for finding matches in a preferred call router [PHO0027443 – 2751]. To the extent WF's routers embody such techniques they would fall within this claim.	
23. The system of claim 15, wherein said speech recognition engine is distributed between a client device and a server system which receives streaming speech data having reduced latency data content before silence is detected and the utterance is complete.	In Wells Fargo's IVR system, the speech recognition engine is distributed between a client device and a server device. For example certain functions such as end pointing and voice activity detection (VAD) are performed by the SR engine component at the client, while the remainder are done on the recognition server. PHO004156; PHO010513; PHO010530; The client portion of the SR engine is configured to send data only while the user is talking (before silence and the utterance being complete) thus reducing the amount of data otherwise associated with the user utterance. [PHO004313 – 4317; 4249 – 4260; 2739 -2742; 4318-4332; 4577 – 4578; 4490 – 4494]	See limitation 9.
27. A method of interacting with a user of a client-server based natural language query system comprising the steps of:	The Wells Fargo interactive voice response (IVR) systems include a natural language query system (hardware and software) that interacts with Wells Fargo customers (users) using the steps of the method described below. WF employs at least three separate IVR systems found to date at 800-642-4720 (banking); 800-368-7550 (Wells Fargo Advantage Funds) and 800-TRADERS (872-3377) (Wells Fargo Investments). While testing has only been done extensively on the banking voice response system, the other systems are believed to perform the steps below as well. See e.g. PHO10434 – 10438 ("Speak your responses and the Automated Phone System will take care of the rest") and PHO10440 (tell the Automated Voice Response System what you	See limitation 1.

	would like to do")	
	Each of the IVR systems uses an automated speaking persona which imitates a human to interact with Wells Fargo customers.	
27.1 a) initiating a first interactive speech session between a client device and a server system on behalf of a user;	When a customer calls the IVR system and uses his/her voice to interact with the system, an interactive speech session is initiated between a client computing device and a speech recognition server. See e.g. PHO PHO002900-04 "The Speech Integration Module—Nuance speech recognition system uses a client-server architecture. The client consists of one or more IVRs, containing all the telephony components used to accept and control a connection from an end user speech recognition begins on the shared resource boards in the IVR The IVR uses the resource manager to dynamically select a recognition server to perform the actual speech recognition processing. "The client and server are shown generally with the arrows above but can include other circuits.	See limitation 1.
27.2 b) providing a database of query/answer pairs coupled to the server system concerning one or more topics	The WF IVRs provide at least one electronic database that includes pairs of questions and associated answers on topic it can cover with customers. For example, as seen in [PHO10425 - 10433] some of the topics covered by the IVR at 800-642-4720 is a customer's balance or an account history. Thus in response to the query for "balance" the system gives an answer that indicates the customer's account balance.	See limitation 1.2

which can be responded to by the natural language query system during said first interactive speech based session with said user;	Sometimes the answer provided by IVR, as soon as she determines the customer's query, is as simple as finding a person who can give the rest of the answer on the topic. See e.g. PHO10427.	
27.3 c) providing a speech recognition engine distributed between the client device and server system, and such that speech data is configured for said first interactive speech session to have a reduced data content to reduce latency, and is streamed to the server before silence is detected;	See above; the SR engine is distributed as seen in the diagram. The Wells Fargo IVRs provide have a speech recognition (SR) engine that meets this limitation as acknowledged in WF materials (see PHO10434: "using advanced voice recognition") and the literature of the various vendors (including Intervoice, Nuance) who supplied the components of the IVR. (PHO010442-51; PHO010464 (Intervoice identified as source of Omvia speech enabled system); PHO010469; PHO010619-626 (Nuance further identified as assisting with system). The speech engine is embodied in software (Nuance 7.x/8.x) which must operate on a computing system referred to as a speech server. [PHO004155 - 4160; 3007; PHO10511 (Omvia Speech Integration Module runs on a recognition server)] The hardware used by Wells Fargo for implementing this is described for operating Nuance 7.x/8.x such as set out in PHO004160; PHO003007; PHO010511 -540]. As noted in other literature, certain functions such as end pointing and voice activity detection (VAD) are performed by the SR engine component at the client, while the remainder are done on the recognition server. PHO004156; PHO010513; PHO010530; The client portion of the SR engine is configured to send data only while the user is talking (before silence and the utterance being complete) thus reducing the amount of data otherwise associated with the user utterance. [PHO004313 – 4317; 4249 – 4260; 2739 -2742; 4318-4332; 4577 – 4578; 4490 – 4494]	See limitation 9.
27.4 wherein said speech recognition engine is	The SR engine recognizes the set of words/phrases spoken continuously by a customer for the query during an interactive speech session. [PHO003007; 2730 – 2731; 4143] This speech engine is embodied in software (Nuance 7.x/8.x)	See limitation 1.1.

adapted to recognize a first set of words and/or phrases received from said client device from a user during said first interactive speech session concerning a first topic taken from said one or more topics; further wherein said first set of words and/or phrases can include a natural language query presented as continuous natural language spoken data;	which must operate on a speech server computer. [PHO004155 - 4160; 3007; PHO10511 (Omvia Speech Integration Module runs on a recognition server)] The hardware used by Wells Fargo for implementing this is described for operating Nuance 7.x/8.x such as set out in PHO004160; PHO003007; PHO010511 -540]. Beyond this acknowledgement in WF's supplier literature, the existence of this SR engine behavior is also shown in the testing protocols provided by Phoenix on the WF system. [PHO10425 - 10433] The fact that the system recognizes words during an interactive session spoken as a natural language query is apparent from the audio clips provided by Phoenix. [PHO10423 - 424] The vocabulary of words which the system is programmed to understand are embodied in a structure referred to as a SR "grammar." The Wells Fargo IVRs use grammars trained to respond to Wells Fargo-specific dialog data, such as Wells Fargo product/service offerings. [PHO010442 - 480; 10619- 10626; 4462 - 4463]	
27.5 c) providing a natural language routine coupled to the server system adapted to process said first set of words and/or phrases and identify a response to said natural language query based on said query/answer pairs;	The Wells Fargo IVRs provide a <i>natural language</i> (NL) routine that meets this limitation as acknowledged in the literature of the various vendors who supplied the components of the IVR. See e.g. PHO002806 "The Nuance System lets you define natural language understanding in your recognition packages. A natural language understanding system takes a sentence (typically a recognized utterance) as input and returns an interpretation—a representation of the meaning of the sentence." See also PHO0002807-2821. This routine is used to determine the meaning of the user's query based on the words/phrases identified by the SR engine. This NL engine is also embodied in software, and typically operates on the speech server as well. In the case of the WF IVR this engine is embodied either as part of the Nuance 7.x/8.x software and/or other natural language software provided by Intervoice. [PHO003320 - 3361; 4473] This routine may also be present in any products used by Wells Fargo for directing calls. See e.g. PHO010632 – 639 "callers interact with an automated speech recognition-based solution that enables them to speak naturally and then to be quickly routed to the correct destination."	See limitation 1.3.

27.6 wherein said natural language routine is adapted to consider only a subset of said first set of word and/or phrases, and further can consider words and/or phrases in said natural language query which are not present in said query/answer pairs to determine said response;	The WF NL routine behavior is also shown in the testing protocols provided by Phoenix. [PHO10425 - 10433] In these routines it can be seen that the NL routine only considers some of the words/phrases (such as the phrase "account history") in the query. Moreover the NL routine can consider other words/phrases (such as "ah" "uhmm" and other dysfluencies, as well as extra words like "give me my") presented by the SR engine output that are not even present in the questions/answers and still determine the response to the query. See e.g. PHO PHO010642 ("With Nuance 8.0, companies can build NVP applications that recognize a variety of speech inputs—from simple "yes/ no" responses to natural sentences like "hi I'd like to check my account balance.") and Say Anything technologies explained at PHO003338 – 3361.	See limitation 1.4.
27.7 d) providing an interactive electronic agent coupled to said natural language routine which is customized for the user and configured to: i. provide a prompt to the user during said first interactive speech based session with suggestions on queries which can be made to the natural language query system concerning said first topic; ii. provide a confirmation of a substance of said natural	The Wells Fargo IVRs provide an interactive electronic agent (as acknowledged in WF's own press releases above) which provides the answer to the customer after it is determined by the SR/NL routines. This electronic agent is presented to customers in the form of an artificial persona that articulates answers in audible form. As seen in testing protocols provided by Phoenix [PHO10425 - 10433] and the audio files [PHO10423 - 10424] the electronic agent provides specific suggestions to callers on what types of queries can be made within the topic of account history (" What type of transaction are you looking for? You can say checks, transfer funds or other services") [PHO10432] As above, note that the agent repeats and confirms the query made by the caller "JNG: Um, I need help transferring funds. WFB: All right, transfer funds." [PHO10432] The agent then goes on to provide a further follow up response: "Would you like to transfer from this account or into this account" [Id.] As seen in the protocols, the electronic agent can continue to give an ongoing series of audible responses directed to WF banking topics until the user decides to terminate the session.	See limitation 1.5

language query;		
language query,		
iii. provide said		
response to the		
user from the natural		
language query		
routine;		
wherein said		
interactive		
electronic		
control agent is		
configured to		
provide		
iterative audible		
responses to user natural		
language		
queries directed		
to said one or		
more topics		
until said		
interactive		
speech based session is		
terminated.		
terminated.		
20 Th	WE's all the side of the side	G. Limitarian 7
28. The method of claim 27	WF's electronic agents are configured with parameters (such as the choice of a female gender	See limitation 7.
further	for the voice) based on the WF's understanding of	
including a	the preferences of its customers. See e.g.	
step:	PHO010444 "provide a business-like, efficient,	
customizing	professional voice that's task and audience	
perception	appropriate" See also PHO010445 – 47;	
related	PHO010452 – 454;	
parameters of the electronic		
interactive		
agent, including		
one of a gender,		
a visual		
appearance,		
and/or voice characteristics		
including one		
of pitch,		
volume and/or		
speed.		
29. The method of claim 27,	The user can conduct a second session to get additional answers on the same topic (for	In the NSRS version 6 system, a user can conduct a second session to get additional answers on the same topic (for

wherein the	example, account history) by calling the IVR	example, account history) by calling the system again. See
electronic	again.	[Kent Decl., Ex. 10 (Nuance6) at WF00000597-604].
interactive		
agent can		See Sharp Decl., ¶ 27.
conduct a		- "
second		
interactive		
speech based		
session with		
said user		
directed to		
continue with		
additional		
queries and		
answers		
concerning said		
first topic.		

KENT DECLARATION EXHIBIT 9

FOR IMMEDIATE RELEASE

Contact:

Bruce Dougherty

NUANCE COMMUNICATIONS

- 415/462-8200--

Julie Sauter
BLANC & OTUS
-415/512-0500

CHARLES SCHWAB AND NUANCE COMMUNICATIONS REDEFINE TELEPHONE-BASED CUSTOMER INFORMATION SERVICES

VoiceBrokerTM Service Leverages Leading-Edge Conversational Transaction Technology to Extend Investment Opportunities to Customers

Menlo Park, Calif. – September 24, 1996 – Charles Schwab & Company, Inc. and Nuance Communications announced today the development and availability of VoiceBrokerTM, the first telephone service using speech recognition technology to provide stock, mutual fund and market indicator information to retail customers.

VoiceBroker was crafted with Nuance's conversational transaction technology, comprised of speech recognition and language understanding capabilities that allow a caller to speak in everyday, conversational English in order to get information, and if desired, engage in business transactions over-the-telephone. VoiceBroker is the result of a Charles Schwab initiative to continuously improve customer satisfaction while reducing costs, and was put in place in conjunction with SRI Consulting.

"Conversational transactions represent the next revolution in customer service," said Ronald Croen, president and CEO of Nuance Communications. "Companies can now turn the telephone into a high-function customer service device that opens up a new range of business transactions to the customer."

"Speech-based systems such as VoiceBroker deliver an exciting new degree of convenience to customers, while allowing companies to reduce costs and operate more efficiently," said Croen.

VoiceBroker in Action

VoiceBroker understands and responds to inquiries on every security listed on the New York Stock Exchange, the American Stock Exchange and Nasdaq system. Customers can simply call up and say the name (or variations of the name) of more than 13,000 stocks, mutual funds or market indicators in order to get up-to-the-minute quote information. Schwab customers can create personal stocklists so customers can get multiple quotes

automatically. VoiceBroker is currently available to customers in a dozen states, and will be rolled out to customers on a nationwide basis in the coming months.

"The Nuance-based system is surpassing all our expectations," said Alan Nathan, vice president of voice technologies at Charles Schwab. "We are already seeing dramatic results, a majority of VoiceBroker customers surveyed saying it's 'easy-to-use, fast and I'd recommend it to a friend."

The Foundation: Nuance's Conversational Transaction Technology

Nuance's conversational transaction technology provides the highest performance speech recognition capabilities available — it can recognize the largest active vocabulary for telephone-based services, combined with a language understanding capability that enables recognition of continuous, natural speech, with meaning comprehension. The product is completely speaker independent, and no training on the part of the user is required. It has also established the highest industry benchmarks for accuracy.

About Charles Schwab

Charles Schwab Corporation is one of the fastest growing financial service companies and the leading provider of discount brokerage services in the world. Schwab capitalizes on technologies that provide customers 24-hour accessibility to trading and investment information.

About Nuance Communications

Nuance Communications focuses on enabling conversational transactions for business services through high-performance software. The company develops advanced technology for speech recognition and language understanding that dramatically enhances an organization's ability to serve its customers effectively and operate more efficiently. Nuance delivers products and services to customers in financial services, banking, travel and telecommunications industries. Venture capital funded in 1995, the company is headquartered in Menlo Park, California.

About SRI Consulting

SRI Consulting, a wholly-owned subsidiary of SRI International, combines strategic business consulting with technology expertise to help companies compete more effectively in changing global markets. Silicon Valley-based SRI International, celebrating its 50th anniversary in 1996, is one of the world's largest research, technology development and consulting firms.

###

TO EDITORS: VoiceBroker is currently available to customers in Alabama, Alaska, Arizona, Arkansas, Colorado, Georgia, Louisiana, Mississippi, New Mexico, Nevada, Oklahoma, Oregon, Texas, Washington, and Wyoming

KENT DECLARATION EXHIBIT 10





Nuance6

The most accurate and flexible speech recognition software commercially available

Nuance6, the core client/server software in the Nuance Communications Conversational Transactions™ product suite, combines unparalleled accuracy and natural language understanding to extend the performance advantage of Nuance in speech recognition applications. Utilizing advanced linguistic and statistical models to interpret and understand natural human speech, Nuance6 enables





sophisticated speech recognition applications which allow users to talk to computers as if they were speaking with human agents. Nuance6's open, scalable software architecture, comprised of the Nuance RecServerTM and the Nuance RecClientTM, makes efficient use of hardware resources to allow the implementation of affordable, flexible speech recognition applications that give users a competitive advantage in today's rapidly changing business environment.

Powerful client/server software architecture

Nuance's distributed architecture allows recognition to happen on a separate stand-alone machine or network of servers, expediting the recognition process and resulting in memory savings because recognition data objects can be shared simultaneously across multiple recognition searches. By dynamically allocating recognition resources with the Resource Manager, Nuance6 capitalizes on the efficiency of processing of multiple recognition requests in parallel and provides automatic fail-over protection in the event of a hardware or software failure. Nuance's client/server architecture also allows applications developers to decouple audio acquisition from recognition processing. The RecClient performs all audio and telephony functions required by speech applications, including echo cancellation and endpointing when DSP is unavailable, and manages the collection and transmission of speech to the multi-threaded RecServer. The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user.

Open and scalable

Speech recognition solutions built with Nuance provide tremendous flexibility and choice because Nuance software is open and fully scalable. While the technology is maximized for telephony applications, it can support many different audio providers on multiple IVR platforms, and its supported availability across multiple UNIX platforms allows users to take advantage of continuing hardware improvements. Nuance6 supports small

and large applications with vocabularies from two to 15,000+ words, and easily accommodates variable capacity requirements. Because the RecClient provides a consistent interface to the application, the application does not have to be recompiled to support different audio providers (like telephones and microphones) on the same IVR platform.

Unmatched recognition accuracy

Nuance's approach to speech recognition achieves a very fine acoustic resolution, and therefore a higher degree of accuracy and a better ability to handle large-vocabulary applications. By partitioning the digitized speech signal into a sequence of short frames, and then using algorithms and statistical processes to analyze the data, the RecServer determines the words or phrases that were spoken. To optimize the accuracy of the RecServer, Nuance uses a unique combination of Hidden Markov Models, a statistical technique that integrates linguistic rules and training from recorded speech databases, and innovative Gaussian-mixture processing.

Natural language understanding

Once the RecServer has recognized the speech, it uses a template-matching natural language approach to determine meaning. This means the RecServer takes the relevant pieces of recognized speech and matches them against a customized template of pre-specified information. This template approach is more robust and sophisticated than word spotting or "tag" approaches and easier and more efficient than an application-specific sentence parser.

TOP

Numico Key Readures

- speech recognition.
- natural language understanding
- telephony control
- cross-machine load balancing
- playback of audio prempts
- DTMF delection:
- audio recording

- leasy-to-use powerful API
- SOL query génération
- automable fall-over protestion
- barge-in
- landline and cellular environments
- Nabest
- confidence scores

TOP





Financial Service Solutions from Nuance

Nuance Communications is dedicated to providing the best speech recognition and speaker verification solutions to financial service companies. Nuance has developed its technology and services, from speaker verification to the grammar update service, with the needs of financial service providers in mind. Customers like Charles Schwab, Lloyd's TSB, and American Express can attest to the fact that Nuance solutions are accurate and effective.

Product and Service Features

- High accuracy on challenging tasks: Nuance provides
 consistently high recognition accuracy for tasks critical to financial
 service applications, including stock and option quotes, natural
 dollar amounts, and natural language commands like "buy 275
 shares of Microsoft at the market."
- Robust operations: Nuance has architected its speech recognition software to have high availability in production environments.
- Nuance Verifier: Nuance has speaker verification functionality
 available as an integrated feature with the Nuance6 speech
 recognition engine. With its high-performance core technology and
 integration with speech recognition, Nuance Verifier provides a new
 level usability, security, and scalability for voice authentication.
 Nuance Verifier adds a high level of security to over the phone
 transactions, and makes customers feel more secure about their
 electronic transactions.
- Preconfigured grammars for common tasks: Nuance has
 developed highly accurate grammars for tasks common to financial
 service applications, including natural dollars, natural numbers,
 stocks and funds, and continuos digits. These grammars have been
 tuned for optimal performance, and save time in application
 development as well as provide high accuracy recognition.
- Stock and Funds Grammar Maintenance: Nuance has a team
 dedicated to keeping the stocks and funds grammar up to date,
 reflecting the latest IPOs, name changes, mergers and delistings.
 Updates to the grammar are available on a weekly basis. In addition,
 customers can track changes to the grammar, as well as make
 requests via the telephone or a secure web site built specifically for
 grammar maintenance subscribers. The maintenance service
 monitors the grammar to ensure high accuracy and complete
 coverage.

TOP

Nuance also has experience developing and building speech applications for financial services. Charles Schwab, Lloyd's TSB, and Visa Interactive have all developed financial applications with Nuance. The experience gained in developing these applications, and also through work done with panel houses and customer acceptance testing, has given Nuance a substantial knowledge base regarding how people perceive and interact with speech enabled applications.

Check out our Stock Quotes Demo.

TOP





Nuance's Better Banking Demo

Nuance Communications' Better Banking demonstration shows the convenience of Nuance's Conversational Transaction technology for personal account management, funds transfer, and bill pay applications.

Listen for These Nuance Advantages

- Natural Grammars: The demonstration shows Nuance's ability to allow natural phrasing of dollar amounts, dates, and transaction types. For example:
 - Dollars can be "twenty-five dollars and thirty-two cents" or "twenty-five thirty-two"
 - Dates can be "June 3rd" or "June 3rd 1996" or "today" or "the first of the month"
- Natural language understanding: Nuance's sophisticated natural language processing accurately determines the correct meaning of naturally phrased commands such as "Transfer from Checking to Savings" or "Transfer three hundred dollars to Savings from Checking".
- Continuous speech: The demonstration supports continuous speech. No artificial pauses are required.
- Accurate Recognition: High accuracy and easy transaction modification are essential for user confidence. By combining superior core recognition technology and friendly interface design, the user can be confident that their transactions will be submitted correctly.
- Speaker independence: The system does not need to be trained for the user's voice, so it's easy to use from the very first call.

TOP

Here's How to Use the Demo

- Dial (650) 847-7438
- Perform any of the following transaction types supported in the demo:
 - Check Account Balance
 - * "What's my savings account balance?" (or checking or VISA accounts)
 - * "Check my checking account balance?"
 - Transfer Money
 - * "transfer money from savings to checking"
 - * "transfer \$325 to checking from savings"
 - Pav Bills
 - * the phone company (or Pacific Bell)
 - * Bank of America (or BofA, or the Mortgage)

- * VISA
- * American Express (or Amex)
- * New York Times (or the paper, or the times)
- * Pacific Gas and Electric (or the power company, or PG&E)
- * say HELP to hear a full list.
- · Review Recent...
 - * checks
 - * withdrawals
 - * deposits
 - * transfers
 - * bill payments
 - * transactions

Press "*" to interrupt the list and return to the menu.

- At any point you can say "HELP" to get more information.
- · Just say "goodbye" to end the demo

TOP





Demo

"My Voice is My Password"

This demo of <u>Nuance Verifier</u> shows just one of the many user interfaces that the technology supports for accurate and seamless speaker verification. Call it with a friend or co-worker to attempt "unauthorized" access.

Call: (650) 847-7452

Enroll: speak your 7 digit phone number and then follow the prompts

Verify: you'll be asked to say "my voice is my password" a few times to be verified

Unique features to notice:

- Integrated verification and speech recognition for a simple and seamless interface: you can speak your phone number to identify yourself and the password phrase to be verified.
- Real-time verification: you can verify your voice immediately after enrollment - there is no need to wait for the software to process your speech and compare it to other users who might be enrolled.
- Dynamic decision making: the system will only prompt for enough verification speech to make a confident decision (rather than requiring you to verify more than necessary).
- Accuracy from any telephone: Nuance Verifier delivers the highest accuracy and allows users to call from any telephone.



Press Room



Nuance's Travel Plan Demo

Nuance Communications is focused on developing products and services for the travel industry. Travel service providers like American Express and British Air have found that Nuance solutions provide a fast and easy way for callers to get information and even book travel. To use the system, simply dial 650-847-7427.

The Nuance Travel Plan demo shows how easy it is for callers to get information about fares and schedules using speech. Simply by answering questions in a normal voice, callers can enter an itinerary and get real information quickly. The demo includes information on 250 airports in the United States and abroad. To use the demo, simply call 650-847-7427.

While you are trying the demo, notice these Nuance features:

- Speaker Independence: For a system to be a success, it has to be able to understand a wide variety of regional, and even international speakers. When British Air piloted a system based on Nuance, they found that over 20% of the callers to the system had foreign accents. Nuance was still able to provide highly accurate recognition, and even foreign speakers could be understood.
- Natural Language: People refer to cities and airports by many different names. Because of this, the system has to be robust to handle wide variations not only in accents, but also in the way people refer to cities. For example, John F. Kennedy International Airport can also be called New York Kennedy, JFK, Kennedy International, and many other variations. Nuance builds these variations into the system so that it is easy for callers to use. When selecting flights from the list offered, callers can say phrases like "the second American flight," or "the seven thirty five flight."
- Barge-In: This demo allows callers to speak when they want to. At any time you can respond to a question, even if the system is speaking. For callers who know how to use the system, barge-in enables even faster transactions.
- Fast and Easy to Use: Getting flight information using touch tones is a slow and frustrating process. Even when callers know the flight number before they call, they still have to work through layers of menus and instructions to get the information they want. Callers who don't know their flight number almost always end up waiting on hold to speak to a customer service agent. With Nuance speech rec, callers get the information they need quickly.

Please let us know if you experience any difficulty with this demo.





Nuance's Stock Quotes Demo

Nuance was the first company to provide speaker independent speech recognition for stock quotes. Charles Schwab & Co. became the first brokerage to offer quotes to their customers in 1996. Since then, Nuance has continued to lead the market, being the first company to provide natural language options quotes, and the first company to provide speech enabled financial transactions.

This demo shows Nuance's ability to provide highly accurate quotes on over 13,000 stocks, mutual funds, and market indicators. To reach the demo, call **650-847-7423**. Simply say the name of the company, fund, or market indicator.

While you're speaking, listen for these Nuance features:

- Large Vocabulary: This system has the largest vocabulary of any speech recognition system. It includes every security on the New York, NASDAQ, and American Exchanges, as well as selected pink sheets, market indicators and thousands of mutual funds.
- Natural Language: Nuance provides unparalleled flexibility in allowing customers to refer to companies. Company names change, and callers often don't know a company's official name. To make the system more usable, Nuance has included many variations on each company name. For example, the Ford Motor Company can also be called Ford or Ford Motors. All told, there are over 2,000,000 ways that callers can correctly ask for a quote from this demo.
- **Barge-In:** This demo allows callers to speak when they want to. At any time you can say a company name, even if the system is speaking. For callers who know how to use the system, barge-in enables even faster transactions.
- Speaker Independence: Nuance6 is speaker independent, and robust in recognizing speakers with foreign accents. Try having other people use the system to see how easy it is for anyone to interact with Nuance speech recognition products.

Please let us know if you experience any difficulty with this demo.





Demos Overview

Nuance's technology is speaker independent, and can be used with virtually no training. Callers merely answer the questions and say what they want to get information and conclude transactions. Nuance demos show Nuance's ability to provide highly accurate, easy-to-use speech recognition interfaces. These applications show how powerful speech recognition can be by enabling applications for which touch tones have proven inadequate.

Try any of them to see what speech recognition can do for you.

Better Banking | Nuance Verifier | Stock Quotes Travel Plan

KENT DECLARATION EXHIBIT 11

CTI Test Drive Page 1 of 6

tmclabs.gif (5633 bytes)	

October 1998



Nuance Conversational Transactions Suite Nuance Communications

1380 Willow Road Menlo Park, CA 94025 Ph: 650-847-0000; Fx: 650-847-7979

Web site:

http://web.archive.org/web/19990209071605/http://www.nuance.com/

Price: Contact vendor.

of any human-computer interaction.

While they were intended as conveniences, early speech recognition applications occasioned more than a little frustration. Indeed, disgruntled users found these application nearly as balky as ordinary, non-speech-enabled auto-attendants. The reason? Auto-attendant, speech-enabled or not, puts all the convenience on the computer side

Inputting information through a menu-driven process creates a sensation much like the one we feel when we sit behind the wheel of an idling car, while we wait for a break in traffic. We fancy the traffic is inspired by some perverse agency bent on vexing us, and we resign ourselves to fitting in however we can, as stressful as that may be.

Fortunately, the old, "fit in" model of speech recognition is giving way to a new approach: natural language recognition. This new approach lets the caller "own the road," as it were. The caller speaks naturally, and the speech recognition application parses and channels the input on behalf of the computer. The computer no longer requires the caller to parse his or her utterances according to some infernal script or menu.

Now, if the new speech recognition lets callers own the road, the

RATINGS (0-5)

Installation: 5
Documentation: 5

Features: 4.5 GUI: N/A Overall: A CTI Test Drive Page 2 of 6

Case 3:08-cv-00863-MHP Do

Document 52-12 Filed 09/08/2008

Page 3 of 7

question remains: Who is going to build the road? Developers who take advantage of advanced speech recognition engines. One such engine, the Nuance Speech Recognition System, is discussed in this review. So, buckle your seatbelts, and let's get started.

INSTALLATION

The Nuance system, which ships on a single CD, installs on selected platforms, including Windows NT, Solaris, AIX, and SCO OpenServer 5. We installed the product on a Pentium 200 MMX computer with 64 Megs of Ram running Windows NT.

The installation involved little more than inserting the CD in the computer's CD-ROM drive. Once we had the CD in place, it ran automatically, so we did not have to execute a setup file. From this point, right until the very end of the installation, the entire operation was self-propelled. The system never once asked for a change in directories or for help in finding any specific Windows files. This ease and rapidity compelled us to give the Nuance system the highest allowable score for installation.

DOCUMENTATION

We received two documents with the Nuance system. One document, a thin booklet, was entitled Getting Started. The other document, a 400-page tome, constituted the Developer's Manual. The Getting Started booklet provided information on the platforms supported by Nuance, as well as the audio setups for those platforms. It also reviewed some issues specific to the Solaris operating system. The remainder of the booklet described the commands necessary to run applications on the system. The Developer's Manual covered many aspects of developing applications, from compiling recognition packages to run-time support to special topics.

We didn't stop at reviewing the literature supplied by Nuance. We also explored the Nuance home page. We noticed that the section of the Nuance site devoted to Nuance6 was quite thorough. In this section, topics such as descriptions, features, supported platforms, the developer's toolkit, and developer training were covered. In addition, new features of the Nuance system, such as the Speech Verifier, were illustrated.

Perhaps the most impressive portion of the site was the Nuance demo section, which includes four demos, which any visitor can access. Visitors who take advantage of these demos can use them to test the accuracy and usability of the system. We encourage anyone to go the Nuance Web site and try these very useful demos.

FEATURES Core Client/Server Software

• Speech recognition, which eliminates the need for complicated

- and frustrating touch-tone call attendants.
- Natural language capabilities, which allow the system to recognize speech patterns rather than just single words. (Thus, the system can pick out key words. If the system fails to recognize a particular word, it does not reject the entire phrase.)
- Telephony control, which allows the Speech Server to control a variety of telephony processes.
- An easy to use, powerful API, which facilitate the creation of speech recognition applications.
- SQL query integration, which gives developers access to a wide variety of database functions.
- Barge-in capabilities, which allow callers to use the system without having to insert unnatural pauses.
- Confidence scores, which indicate the reliability of the matches the system makes between the input it receives and the words in its vocabulary. (Confidence scores are configurable to create systems geared towards speed or accuracy.)

Nuance Verifier

- Speech recognition as a means of instituting security.
- Simultaneous recognition and authentication of speech, which allows for very fast, real-time verification of speakers.

Developer's Toolkit

- Grammar specification.
- Natural language specification.
- SpokenSQL, which can be used to generate a database query from speech.
- Xwavedit, which is a means of recording prompts. (Allows the developer to edit prompts for maximum recognition accuracy and efficiency.)

OPERATIONAL TESTING

Although actually developing our own application was beyond the scope of our review, we acquainted ourselves with the Nuance systems by running a sample application, and by reviewing a few of the demonstration programs we found on Nuance's home page.

Sample Application

This application, working from a specified vocabulary, prompts the user for input, compares the input to the words in the vocabulary, and returns the word that it deems the closest match, along with a confidence score. We experimented quite a bit - so much, in fact, that we started to feel at one with our microphone-equipped headset. In the course of our work, we took careful note of the confidence scores, which displayed such variation that we satisfied ourselves

that the system was acting properly.

Demonstration Programs

Two of the demonstrations, Travel Plan and Better Bank, gave us a chance to evaluate Nuance's natural language capabilities. Also, Travel Plan let us check out Nuance's barge-in feature. Another feature, speech verification, was displayed to advantage in the Nuance Verifier demo. And, finally, the Stock Quotes demo let us work with a system that boasted an enormous vocabulary.

Travel Plan: This demo, which was an interactive, real-time travel planner, showed off how a Nuance-based application could recognize alternative names for airports. (We need hardly add that the ability to cope with alternative names is one of the manifestations of natural language recognition.)

We started by referring to an airport first as JFK, then as Kennedy. The system responded appropriately, recognizing that both names corresponded to the same airport. So, we decided to try something a little trickier. We decided to refer to Dulles, for Dulles International Airport. We supposed the system might mistake Dulles for Dallas, which sounds much the same. We supposed wrong, however. The system did in fact recognize Dulles, and we had to admit we were impressed.

Before we moved on to the next demo, we made sure to check out the system's barge-in capability, that is, the system's ability to accept user input even if the user supplies it before hearing the appropriate prompt. This capability has its advantages. It lets callers familiar with the menu to move through it more quickly. However, it can increase the potential for errors. Hence, good barge-in functionality accommodates impatient callers without sacrificing accuracy.

Since we're naturally impatient, we had no difficulty testing the barge-in functionality. For example, after we had heard only two of three flight options, we knew which option we wanted. So, we barged in, not caring to hear the third option. The system responded exactly as it should have. It stopped reading the flight information, and it asked us if we would like to obtain pricing information on that particular flight.

Better Bank: In this demo, Nuance's Natural Grammar and Natural Language capabilities are displayed. These capabilities go beyond the recognition of individual words. Instead, the idea is to recognize alternative word combinations, sparing the user the challenge of speaking information in any particular predefined format.

The few tests we tried here gave us favorable responses. When asked how much we wanted to transfer from checking into savings, we responded, "Fifty-seven dollars and thirty-two cents." Later, in

Page 6 of 7

response to the same question, we said, "Fifty-seven thirty-two." In both cases, the demo transferred the correct amount.

We did have one problem with the demo, however. When we indicated we wanted to make a payment on our American Express bill, the system asked us to indicate the amount. We said, "Pay in full." However, the system told us it didn't understand. Then, we tried other, equivalent phrases, to no avail.

Finally, we tried a question: "How much do I owe?" The system then asked us if it was correct that we wanted to pay twelve dollars. Well, that wasn't the balance. Perhaps "twelve" was the closest match the system could produce. If so, it would appear the demo's programming simply didn't anticipate the sort of input we provided. We don't suppose, however, that our problem had anything to do with the underlying speech recognition engine.

Nuance Verifier: We called into the demo and enrolled by giving our seven-digit phone number. The system then asked one of our engineers to repeat the phrase "My voice is my password" three times until it was satisfied that it could recognize his speech patterns.

Then, to proceed with the demo, we had a second engineer attempt to log into the first engineer's account. When the system prompted the second engineer to say, "My voice is my password," he complied, but the system refused to let him access the account. Thus, we confirmed the new speech verification feature actually worked.

Stock Quotes: Finally, we came to what is probably the best-known application of the Nuance system, the Stock Quotes demo. In 1996, Nuance developed a system for Charles Schwab & Co. This system, which lets the company offer stock quotes to its clients, needs an enormous vocabulary, for there are thousands of stocks clients might ask about. In fact, there are over thirteen thousand stocks, mutual funds, and market indicators.

To make matters even more complicated, many of these stocks may be referenced in multiple ways, all of which have to be recognized by the system. That such a large system should work so well is impressive, for as more words are added into any system's vocabulary, the confidence levels for any matches delivered by the system invariably decline. Thus, in such a system, it is often a good idea to read the caller's input back to the caller, so the caller knows whether the information the system provides really is pertinent to the original input. Also, it might help to read back some ancillary information. For example, in the case of a company name, it might help to read back the company's city and state, the better to distinguish any given company from sound-alike companies.

ROOM FOR IMPROVEMENT

CTI Test Drive Page 6 of 6

It appears Nuance believes (and rightly so) that its role is to provide the basis for voice-based systems, and that creating working speech recognition applications is up to developers. Thus, Nuance concentrates on refining its speech recognition engine. And, while Nuance doesn't neglect to give developers some guidance (with the Developer's Toolkit, for example), it hasn't gone so far as to release a full application generator. We would like to see a development tool of this sort specifically aimed at creating a Nuance-compliant application. Such a tool would be a convenience to developers, and it would (more than incidentally) promote Nuance's interests.

Nuance could look after its own interests in yet another way, and yet again extend a convenience to other parties, by eliminating whatever problem caused our difficulties with the Banking demo. (We suppose the problem is a limited vocabulary.) Of course, this problem doesn't raise any issues with the speech recognition engine itself. We just feel the demo should do justice to the engine. So, you might consider our suggestion a backhanded compliment.

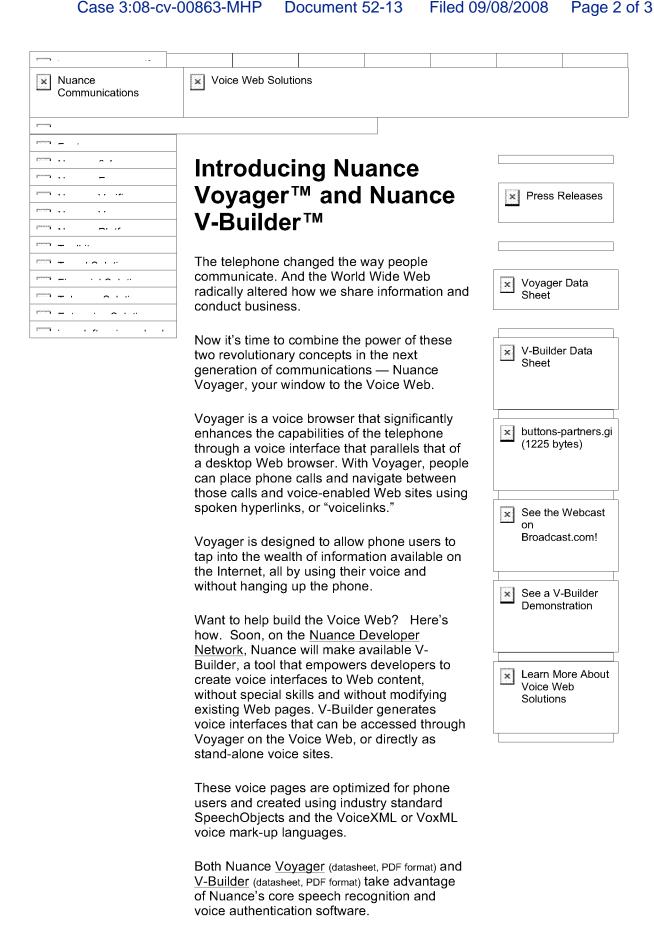
CONCLUSION

The continued evolution of speech recognition seems assured. This evolution is driven, of course, by the need for more natural human-computer interfaces. With the right interfaces, people will no longer need to adapt to the computer's way of working. Instead, it will be the computer that does the adapting; interfaces will acquire whatever attributes are needed to maximize user convenience. One new attribute that is already facilitating human-computer interactions is, as we've seen, natural language recognition. This attribute will soon enhance many applications, thanks to tools such as those provided by Nuance and other companies.

KENT DECLARATION EXHIBIT 12

Document 52-13

Filed 09/08/2008



Nuance provides the industry's most accurate

and scalable speech servers capable of handling millions of calls per day.

Together, these products make possible a new class of enhanced corporate and consumer services, all accessed over the most ubiquitous communications medium on the planet — the telephone.

Welcome to a new and better way to communicate.

Welcome to the Voice Web — enabled by Nuance — built with V-Builder and navigated with Voyager.

The leader in speech technology for V-Commerce and telecommunications

Copyright Nuance Communications 1999. All rights reserved.

KENT DECLARATION EXHIBIT 13

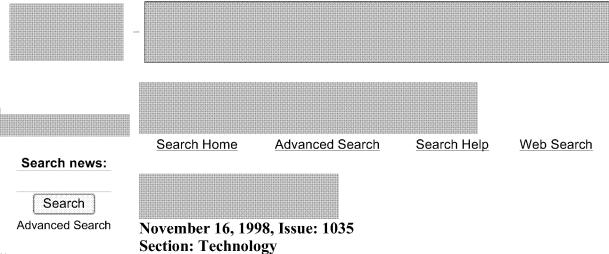
TechSearch Page 1 of 2

Case 3:08-cv-00863-MHP

Document 52-14

Filed 09/08/2008

Page 2 of 3



News

Technology Stocks & Finance

Voice recognition added to Web-based businesses

R. Colin Johnson

Byte.com
CMPmetrics
Data
Communications
File Mine
InformationWeek
InternetWeek
Network Computing
Planet IT
TechShopper
TechWeb News
Tele.com

Menlo Park, Calif. - Nuance Communications recently teamed with Motorola Inc. and Visa International to supplement Web-based commerce with speech-recognition capabilities. The technology will allow Web-based businesses to add integrated speech-recognition access to their services over ordinary phone lines.

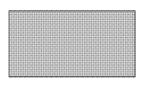
Three technologies were blended-Nuance speech objects, Motorola's Voice Markup Language (VoxML) and a Java application programmer interface-to allow users equal access to businesses from either the Web or the telephone.

"This alliance enables users to conduct business over the Web, such as booking a plane reservation, and then access the same information over an ordinary telephone, such as changing that reservation while you are on the road," said Ronald Croen, president and chief executive officer of Nuance Communications.

E-mail Newsletters Send Feedback

Winmag.com

Nuance speech objects enable developers to assemble applications from a set of reusable speech-application components. Together with the Java API, developers can quickly build applications, in C++, Java or Visual Basic, that permit user access from either phone or Web. Motorola's VoxML simplifies the telephone access to Web-site business with a set of hypertext markup language (HTML) extensions that are used to integrate speech-recognition access to information collected from Web pages.



Free E-mail Login - Sign Up Now

Nuance speech recognition is based on a speaker-independent technology that has been pretrained on a broad cross-section of native speakers of several languages. Nuance has collected and trained its acoustic model using a combination of traditional hidden-Markov and neural technologies. The various accents and pronunciations characteristic of all major geographical regions were represented in one

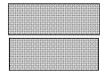
TechSearch Page 2 of 2

Case 3:08-cv-00863-MHP Do

Document 52-14

Filed 09/08/2008

Page 3 of 3



Privacy Statement

training set. For instance, the American English model recognizes speech equally well whether the native is from Boston or the deep south. Nuance speech objects incorporate all accents into a single model for each language.

At the application-development level, Nuance gives developers a single-recognition output and a ranked set of its best guesses at the word that was spoken. Each guess is ranked, enabling applications to match the guesses against information in existing databases. Nuance technology parses grammar as one speaks, so that the meaning of what is said is sent to the application, regardless of the specific natural-language phrase used. Nuance also validates the identity of specific speakers without the need for passwords.

"Our technology can recognize your specific voice print and validate that you are who you say you are without the need for you to speak any specific phrases-just use natural language and we recognize who you are while you are conducting your business," said Croen.

Nuance's speech engine automatically generates confidence scores for each recognition, and sends those scores along to the application too. This lets the application determine whether the user is "on the same wavelength" as the application. For instance, if the application is expecting to hear a number, and the user instead said, "What should I do now?" the recognition engine might pick "four" as the closest number. But the low confidence score would enable the application to realize the user is lost and needs to be prompted in more detail.

Copyright? 1998 CMP Media Inc.

Sponsored Supplements From CMPnet

Why IT has more influence on core businesses than ever.

Get answers to business questions at a Planet IT roundtable.

- The latest incarnation of MS Exchange still lacks features.
- Should Win2000 fear UNIX? If it cares about scalability and reliability, yes.
- Disasters aren't so bad -- if you can recover from them.
- Worried about intrusions? Find out what the hackers already know.



KENT DECLARATION EXHIBIT 14

Nuance 6

Nuance Communications 1380 Willow Road Menlo Park, CA 94025 Phone 650.847.0000 Fax 650.847.7979

Nuance 6, the core client/server software in the Communications product Nuance combines unparalleled accuracy and natural understanding to extend performance advantage of Nuance in speech recognition applications. Utilizing advanced linguistic and statistical models to interpret and understand natural human speech, Nuance 6 sophisticated speech recognition applications that allow users to talk to computers as if they were speaking with human agents. Nuance 6's open, scalable software architecture, comprised of the Nuance RecServer TM , Nuance Resource Manager TM and the Nuance RecClient™, makes efficient use of hardware resources to allow the implementation of affordable. flexible speech recognition applications that give users a competitive advantage in today's rapidly changing business environment.

Powerful client/server software architecture

Nuance's distributed architecture recognition to happen on a separate stand-alone machine or network of servers, expediting the recognition process and resulting in memory savings. This is achieved by simultaneously sharing recognition data objects that can be shared across multiple recognition searches. By dynamically allocating recognition resources with the Resource Manager, Nuance 6 capitalizes on the efficiency of processing of multiple recognition requests in parallel and provides automatic fail-over protection in the event of a hardware or software failure. Nuance's client/server architecture also allows applications developers to decouple audio acquisition from recognition processing. The RecClient performs all audio and telephony functions required by speech applications, including echo cancellation and endpointing when DSP is unavailable, and manages the collection and transmission of speech to the multi-threaded RecServer. The RecServer itself handles speech recognition and understanding. This allows the application to focus on call-flow and dialog management, resulting ultimately in a more seamless interaction with the end user.

Open and scalable

Speech recognition solutions built with Nuance 6 provide tremendous flexibility and choice because Nuance software is open and fully scalable. While

the technology is maximized for telephony applications, it can support many different audio providers on multiple IVR platforms, and its supported availability across multiple UNIX and NT platforms allows users to take advantage of continuing hardware improvements. Nuance 6 supports small and large applications with vocabularies from two to 15,000+ words, and accommodates variable capacity requirements. Because the RecClient provides a consistent interface to the application, the application does not have to be recompiled to support different audio providers (like telephones and microphones) on the same IVR platform.

Unmatched recognition accuracy

Nuance's approach to speech recognition achieves a very fine acoustic resolution, and therefore a higher degree of accuracy and a better ability to handle large-vocabulary applications. By partitioning the digitized speech signal into a sequence of short frames, and then using algorithms and statistical processes to analyze the data, the RecServer determines the words or phrases that were spoken. To optimize the accuracy of the RecServer, Nuance uses a unique combination of Hidden Markov Models, a statistical technique that integrates linguistic rules and training from recorded speech databases, and innovative Gaussian-mixture processing.

Natural language understanding

Once the RecServer has recognized the speech, it uses a template-matching natural language approach to determine meaning. This means the RecServer takes the relevant pieces of recognized speech and matches them against a customized template of pre-specified information. This template approach is more robust and sophisticated than word spotting or "tag" approaches and easier and more efficient than an application-specific sentence parser.

Nuance 6 Key Features

- Continuous Speech Recognition
- Natural Language Understanding
- Barge-In
- Open, Client/Server Architecture
- Machine Load Balancing and Fault-Tolerance
- Landline and Cellular Environments

- N-Best Results And Confidence Scoring
- Multiple Language Support
- Easy-To-Use, Powerful APIs
- Minimal Recognition Latency
- Telephony Control
- Portable Across Major IVRs

Nuance 6 Key Benefits

- Rapid Development of Powerful, Highly Scalable Applications
- Speaker-Independent, Speaker-Trained, or Both Recognition Methods Combined
- Flexible Dialog Design Options
- Ability to Hot-Swap Hardware And Grammars
- Large Vocabulary Support

- Proven, Unmatched Recognition Accuracy in Various Acoustic Environments
- Robust Telephony Capabilities

Nuance 6 Supported Platforms

PLATFORM	AIX	Solaris	Solaris	WinNT	OSF	SCO OS5
	RS/600	Intel	Spare	Intel	DEC Alpha	Intel
Dialogic with Antares	I	√	1	√	√	✓
NMS Audio Provider	√	√	√	√	√	✓
Periphonics			√			
Direct Talk	√					
Aspect/Voicetek			√		√	√
Syntellect				I		

About Nuance

Nuance Communications develops speech recognition, language understanding and speaker verification software to automate access to information and services over-the-phone. Nuance's products enable a user to speak to a computer over the telephone in everyday, conversational language. Headquartered in Menlo Park, California, the company focuses on customer service applications in call centers, particularly within the financial services and travel industries, and on enabling enriched functionality and new services in telecommunications networks. For more information, see Nuance's Web site at http://www.nuance.com or call (650) 847-0000.

KENT DECLARATION EXHIBIT 15



Enabling Software Technologies for Antares

from DSP Software Engineering, Inc



Use **Dialogic Direct Connect**TM to have a Dialogic Sales Agent call SALES SUPPORT you back to discuss this product or any other sales related issue.

DSP Software Engineering, Inc (DSPSE) is an industry leader in software solutions for computer telephony, telecommunications, digital wireless, videoconferencing, secure communications, and consumer electronics. Since 1989, DSPSE has been providing an international software clientele with custom, turn-key software solutions and professionally engineered algorithms for digital signal processing (DSP) based products. DSPSE's software provides easy access to DSP, allowing OEMs and manufacturers of telecommunications products to take full advantage of complex DSP technology through a range of simple software solutions.

DSPSE's PRODUCTS AND SERVICES FOR ANTARES:

DSPSE is unique among the Dialogic Corporation providers. Rather than offering a specific high level product, DSPSE offers the means to create custom products based on DSP technology with the Antares platform. DSPSE has 21 algorithm products that are compatible with the Antares platform and which can be used to quickly build high performance applications that fit specific customer requirements.

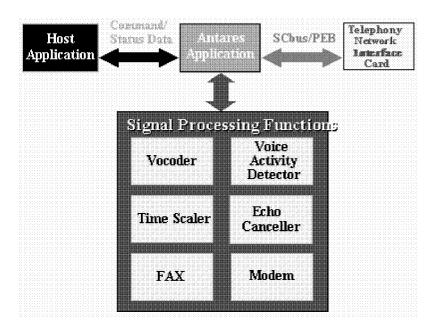
The Systems Group at DSPSE has used these algorithm products to build "private label" solutions on the Antares platform. The combination of Antares specific experience and expertise in the development of DSP algorithms and applications enables DSPSE to quickly create the exact DSP based solution your application requires. These solutions can be built out of any combination of DSPSE algorithms, algorithm products from other sources, and from scratch development to particular specifications.

APPLICATION AREAS:

- Custom Modem and Fax Servers
- All Telephony Related Voice Compression
 - Call Logging
 - Voice Mail
 - International Callback
 - WAN based Voice Calls
- Wireless Services

Page 3 of 4

- Interactive Voice Response
- Audio Conferencing



DSPSE OFF-THE-SHELF PRODUCTS:

Voice and Audio Compression

TIA IS-54 VSELP

DSPSE Low Memory CELP

DSPSE Multi-Rate CELP

DSPSE VoiceWave

ITU G.728 LD-CELP

ITU G.726 ADPCM

ITU G.723 CELP

ITU G.722 SB-ADPCM

ITU G.711 m-law/A-law compander

FAX and Modems

ITU V.22bis/Bell212A Modem ITU V.32bis Modem ITU Group 3 Fax Bell 103 Modem DSPSE V Series Modem Interface

Telephony

IDEC-II Acoustic Echo Canceller
Telco Call Progress Generator/Detector
Telco DTMF Signaling Generator/Detector
Bellcore Caller ID Detector
ITU G165-T Audio Line Echo Canceller
DSPSE Audio Line Hybrid Echo Canceller
DSPSE Speech Time Scaler and Multi-rate Filter

DSP Software Engineering, Inc (DSPSE)

Page 4 of 4

Document 52-16

Filed 09/08/2008

175 Middlesex Turnpike Bedford, MA 01730 USA Phone: 617-275-3733 Fax: 617-275-4323

Email: info@dspse.com

Web: http://web.archive.org/web/19991002214849/http://www.dspse.com/

00-3435-001 3-12-97

| Home | Search | Site Map |

| FeedBack |

© 1999 by Dialogic Corporation

North America (Parsippany, NJ) 1-800-755-4444 or

1-973-993-3030

fax: 1-973-631-9631

(Brussels) 32-2-712-4311 fax: 32-2-712-4300

Europe

http://web.archive.org/web/19991

http://web.archive.org/web/19991002214849/http://www.dialogic.com/forms/salefeed.htm



Dialogic World Headquarters ? 1515 Route Ten ? Parsippany, NJ 07054 ? USA phone: 1-973-993-3000 ? fax: 1-973-993-3093 ? fax-on-demand: 1-800-755-5599 or 1-973-993-1063

Nuance 6

Nuance Communications is the leader in speech technology for V-Commerce and communications transactions. Nuance 6, the company's flagship speech recognition engine, delivers the recognition accuracy, scalability and robustness demanded in enterprise-class natural language speech applications.

Proven PerformanceNuance 6 systems ar

Nuance Communications

1380 Willow Road

Menlo Park, CA 94025

www.nuance.com

Phone 650.847.0000 Fax 650.847.7979 Nuance 6 systems are deployed at numerous Fortune 500 customer sites enabling a range of mission critical applications. These systems currently handle over one million transactions per day around the world.

Nuance 6 customers include:

Charles Schwab and Co. Sears, Roebuck and Co. UPS American Airlines Lloyd's TSB Bell Atlantic Telia

Unmatched Recognition Accuracy

Nuance 6's definitive recognition approach ensures the highest accuracy rates. To optimize the accuracy of the server, Nuance 6 uses a unique combination of Hidden Markov Models, a statistical technique that integrates linguistic rules and training from recorded speech databases, and innovative Gaussian-mixture processing. Accuracy rates in deployed Nuance natural language understanding applications commonly exceed 96%.

Secure V-Commerce

Nuance 6 is integrated with Nuance Verifier to provide the first integrated speaker verification, natural language speech recognition system. The

combination of the two technologies ensures a level of user convenience and security that was previously unattainable. Users are recognized and authenticated simultaneously so that when they say their name, for example, their identity is verified at the same time. This shortens the overall call duration and also eliminates the need for users to remember pin numbers.

Scalable Distributed Architecture

The Nuance distributed software architecture supports stand-alone or networked server configurations, providing cost effective and flexible deployment options. When multiple simultaneous requests are received, they are dynamically allocated to available recognition processes, enabling scalability and automatic fail-over protection.

Standards-based APIs

Java and ActiveX APIs enable developers to create applications using the powerful languages and tools with which they are already familiar. These APIs also enable faster integration with COM or Java-based enterprise and e-commerce servers, and facilitate the development of V-Commerce applications.

Intelligent Natural Language Interface

Systems deployed using Nuance 6 take advantage of Nuance's natural language understanding engine that enables users to speak naturally as though they are speaking to a person. The usability of Nuance applications is enhanced by advanced features, such as single phrase correction, that ensures users do not need to repeat an entire sentence if only part of what they said is not recognized.

Personalized Voice User Interface

To provide a more convenient user experience, Nuance 6 enables users to personalize applications on-the-fly. For example, in a bill payment application, a user may set up their credit card provider as a payee by providing the card type, bank name and account number, along with a personalized word to avoid reentering the information in the future. The convenience and simplicity of this functionality is extremely useful in applications such as voice-activated dialers, personal assistants, and personalized stock portfolios.

BEFORE: "Pay Nations Bank of Delaware VISA account number 1564789876678585"

AFTER: "Pay VISA"

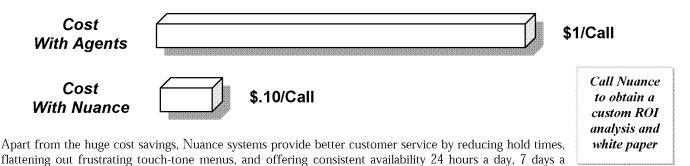
Nuance, V-Commerce, SpeechObjects and Nuance Verifier are trademarks of Nuance Communications. All other trademarks are the property of their respective owners.

© 1999 Nuance Communications. All rights reserved.

DS015-0299

Better Customer Service at a Fraction of the Cost

Nuance software provides better customer service at a significantly lower cost than live agents. While the cost of agents can vary depending on the complexity of the tasks being performed, basic service costs are at least \$1 per call. Equivalent Nuance software, hardware and services cost only \$.10 per call. As a result of these tremendous savings, Nuance customers have recouped their investments within three months of deploying Nuance-based systems.



Other Key Features

week.

- Continuous speech recognition
- Speaker independent recognition
- Support for mixed initiative and directed dialogs
- Recognition of accents and dialects for each language
- Barge-in so users can speak over out-going prompts
- N-best results and confidence scoring
- Minimal recognition latency
- Telephony control

Nuance 6 Supported Platforms, Languages and Interfaces

Operating systems:	NT, Sparc Solaris, x86 Solaris, AIX, DEC UNIX, SCO
IVR Platforms:	Periphonics, Edify, Syntellect, IBM Direct Talk, Brite, Aspect
Audio Interfaces:	Dialogic, NMS, Desktop/Sound Blaster, custom
Available Languages:	American English, UK English, American Spanish, Japanese, German, French
	Canadian, Australian English, and Swedish. Brazilian Portuguese will be
	available in the first half of 1999.
Software Interfaces:	C/C++, Java, ActiveX

About Nuance

Nuance Communications provides enterprise-level speech recognition and speaker verification software to automate V-Commerce and communications transactions. Users interact with deployed Nuance systems over one million times per day using everyday conversational language. The company's target markets include the travel, financial services, and telecommunications industries. Nuance's products support a variety of languages including U.S. English, U.K. English, Australian English, German, Japanese and Latin American Spanish. Nuance was named one of Upside's Hot 100 Private Companies and Red Herring's Top 50 Privately Held Companies. The company is headquartered in Menlo Park, California and maintains subsidiaries and offices in France, Germany and the United Kingdom. For more information, see Nuance's Web site at www.nuance.com or call (650) 847-0000. For more information on V-Commerce see www.v-commerce.com or call 1-877-TRY-VCOM.

ProQuest

<u>Databases selected:</u> Multiple databases...

Nuance Ushers in a New Age of Communications With Voyager Voice Browser

PR Newswire. New York: Oct 5, 1999, pg. 1

Abstract (Summary)

NEW YORK, Oct. 5 /PRNewswire/ -- Nuance, the leader in natural speech interface software for telecommunications, enterprise and Web- based systems, today debuted Nuance Voyager(TM), a voice browser that fundamentally changes the way people use the phone to communicate, access information and conduct business. Voyager significantly enhances the capabilities of the telephone network in the same way that Web browsers widened the appeal of the Internet.

With Voyager, people can place phone calls and reach speech applications or "voice sites" on a network called the "Voice Web." Voyager's voice interface provides a consistent way to navigate the Voice Web with spoken hyperlinks, or "voicelinks" and easy-to- remember commands. In addition, Voyager's personalization capabilities make the telephone more convenient by streamlining access to favorite phone numbers and voice sites.

Voyager is built on Nuance's market leading speech recognition and voice authentication software. The powerful and easy-to-use Voyager voice user interface is rendered through a combination of SpeechObjects and the VoxML (or the forthcoming VoiceXML) language and is based on many years of Nuance experience designing speech applications. Not only can Voyager access the full range of functionality in any application that is built with SpeechObjects or VoxML, but it can also access all other speech applications.

Full Text (1090 words)

Copyright PR Newswire - NY Oct 5, 1999

Product Provides Universal Access for Communications, Content and Commerce;

World's Largest Technology Companies Announce Support

NEW YORK, Oct. 5 /PRNewswire/ -- Nuance, the leader in natural speech interface software for telecommunications, enterprise and Web- based systems, today debuted Nuance Voyager(TM), a voice browser that fundamentally changes the way people use the phone to communicate, access information and conduct business. Voyager significantly enhances the capabilities of the telephone network in the same way that Web browsers widened the appeal of the Internet.

More than 25 leading telecommunications and Internet companies have agreed to work with Nuance and its Voyager product including British Telecom, BroadVision, Cisco Systems, Excite@Home, Intel, Motorola, Oracle, Siebel Systems, Sun Microsystems, Telcordia Technologies and Travelocity.com.

With Voyager, people can place phone calls and reach speech applications or "voice sites" on a network called the "Voice Web." Voyager's voice interface provides a consistent way to navigate the Voice Web with spoken hyperlinks, or "voicelinks" and easy-to- remember commands. In addition, Voyager's personalization capabilities make the telephone more convenient by streamlining access to favorite phone numbers and voice sites.

"Voyager is the first revolutionary improvement to the telephone network since the introduction of touch tone in the 1970s," said Ronald Croen, president and CEO of Nuance. "The combination of Nuance's powerful natural language speech technology and voice interface expertise, the ubiquity of the phone and the vast resources of the Internet will spur new growth and opportunities for telecommunications providers and Web businesses alike."

"For several years, the Web and the telephone network have served separate purposes and catered to different audiences," said Megan Gurley, telecommunications analyst for the Yankee Group. "Nuance Voyager brings the best of both worlds together. It has the potential to not only radically alter the way carriers do business, but also to expand the already burgeoning electronic commerce market."

Voyager is built on Nuance's market leading speech recognition and voice authentication software. The powerful and easy-to-use Voyager voice user interface is rendered through a combination of SpeechObjects and the VoxML (or the forthcoming VoiceXML) language and is based on many years of Nuance experience designing speech applications. Not only can Voyager access the full range of functionality in any application that is built with SpeechObjects or VoxML, but it can also access all other speech applications.

Voyager will provide telecommunications carriers with a platform for increasing the range, quality and profitability of services. With competition for customer loyalty at its peak, carriers are seeking new ways of increasing customer retention. By offering Voyager services through their networks, they can offer a truly intelligent, personalized network interface, a series of enhanced "voice portal" services and commerce capabilities to the mass consumer market.

"Nuance Voyager is a groundbreaking concept that will have significant impact in the industry," said Jeremy Stafford, general manager of Voice Solutions for British Telecom. "It fits with British Telecom's vision, and can become the basis for providing information and services through natural language speech technology. During the next year we plan to work with Nuance in the development, testing and deployment of Voyager-based capabilities for our customers who are looking for competitive advantage in e-commence applications."

Voice Web and Voice Sites -- Voice-Enabled Content and Commerce

E-businesses will be able to dramatically extend the reach of their products and services by using Voyager. For Webbased companies, Voyager offers a new way to serve up content to and do business with the owners of the more than one billion telephones worldwide, an audience roughly 10 times greater than those with connected PCs.

"We're thrilled to be working with Nuance on the world's first true voice browser," said Craig Donato, senior vice president, communities, search & network programming, Excite@Home. "The ability to deliver Web-based content -- along with a host of new voice portal services like Excite voice mail -- is a tremendous benefit to telephone and online users around the world."

In conjunction with Voyager, Nuance also announced V-Builder(TM), a groundbreaking tool that allows users to quickly build voice interfaces to Internet sites and speech applications that can be accessed through the Voyager browser. For more information on V- Builder, please see today's release entitled, "Nuance V-Builder makes it easy to provide voice access to web sites over the telephone."

Voyager Features

Voyager offers many navigation and personalization features that enhance the user experience including:

- -- Place calls simply by speaking a phone number or name
- -- Navigate between voice sites using spoken hyperlinks or "voicelinks" to

access everything from voice portals to corporate intranet

- -- Access multiple voice sites and place several calls without hanging up
- -- Store frequently used information in bookmarks and profiles
- -- Conduct secure transactions

Broad Support for Voyager

More than 25 leading Internet and telecommunications companies have agreed to work with Nuance and its Voyager product including: @Motion, Advanced Communications Group, Inc. (WorldPages.com), Authentix, British Telecom, BroadVision Corp., BeVocal, Inc., Cap Gemini Telecom and Media, Charles Schwab and Company, Cisco Systems, Edify Corp., Energis, Ernst and Young, Excite@Home, Fidelity Investments, General Magic, Inc., Intel Corporation,

Loading "Document Wearse 3:08-cv-00863-MHP Document 52-18 Filed 09/08/2008 Page 4 of 4 7/17/08 8:55 AM

iVillage.com's Astrology Channel (Astrology.net), Mapquest.com, Inc., Merrill Lynch, Intelligent Information, Inc., Mitel Corp., Motorola, Inc., Oracle Corp., Siebel Systems, SmartRoute.com, Sun Microsystems, Talk2.com, Telcordia Technologies, Telia Telecom, The Weather Channel(R) Web site (www.weatherchannel.com), Travelocity.com, UAccess, UpShot.com and Webley Systems.

Pricing and Availability

Voyager will be available to select partners in Q4 1999 with general availability expected in the first half of 2000. Nuance V- Builder for building voice sites will be available in Q4 1999. Pricing has not been announced.

About Nuance

Nuance is the leader in natural speech interface software that makes access to information, transactions, and services over the telephone convenient and secure. Every day, millions of people place calls, make travel reservations, trade stocks, and interact with other telecommunications, enterprise and Web-based systems running Nuance's speech recognition, language understanding, and voice authentication software. American Airlines, Bell Atlantic, Charles Schwab, The Home Shopping Network, Lloyds TSB, Sears, United Parcel Service, and many other leading companies, use Nuance software to provide better customer service while dramatically lowering costs. Nuance is a founding member of the V-Commerce Alliance and has led the industry in the creation of open standards for speech application development. Nuance is headquartered in Menlo Park, Calif. with global sales offices and partners supporting multilingual solutions around the world. To experience Nuance's state-of-the-art speech technology, call 888-NUANCE-8 or visit the company's Web site www.nuance.com.

NOTE: Nuance, Nuance Voyager, Nuance V-Builder are trademarks of Nuance. All other trademarks are properties of their respective owners. SOURCE Nuance

[Reference]

Message No: Industry: COMPUTER/ELECTRONICS; TELECOMMUNICATIONS;

Indexing (document details)

Dateline: New York

Publication title: PR Newswire. New York: Oct 5, 1999. pg. 1

Source type: Wire Feed ProQuest document ID: 45333409

Text Word Count 1090

Document URL: http://proquest.umi.com.ezproxy.apollolibrary.com/pqdweb?did=45333409&sid=-1&Fmt=3&cli

entId=13118&RQT=309&VName=PQD

Copyright © 2008 ProQuest LLC. All rights reserved.



<u>Databases selected:</u> Multiple databases...

Nuance V-Builder Makes It Easy to Provide Voice Access to Web Sites Over The Telephone

PR Newswire. New York: Oct 5, 1999. pg. 1

Abstract (Summary)

NEW YORK, Oct. 5 /PRNewswire/ -- Nuance today unveiled Nuance V- Builder(TM), a groundbreaking tool which enables developers to easily design voice sites by creating voice interfaces to Web sites and other applications. Together with Nuance Voyager(TM), a revolutionary voice browser and Nuance's market-leading speech recognition and voice authentication software, V-Builder will enable any company to provide information and sell products in an automated fashion via an ordinary telephone.

V-Builder is unlike any other development tool because it enables developers to create voice sites in a few hours without speech recognition or telephony skills and without modifying Web pages. These voice sites can be accessed through Voyager and can be interconnected on an emerging "Voice Web" through spoken hyperlinks or "voice links." The tool also can be used to quickly develop voice interfaces for any other application for access over the telephone.

V-Builder uses a combination of Nuance's SpeechObjects, a set of standards-based reusable application components, and the VoxML (or the forthcoming VoiceXML) programming languages, to generate voice interfaces that are accessible directly via the telephone or through the Voyager voice browser. The tool is designed so that savvy speech developers can easily integrate their applications with the Web, and Web developers can easily extend their content for access over the telephone.

Full Text (777 words)

Copyright PR Newswire - NY Oct 5, 1999

Breakthrough Tool Enables Developers to Build Voice Sites Powered by Speech

Recognition & Voice Authentication Technology

NEW YORK, Oct. 5 /PRNewswire/ -- Nuance today unveiled Nuance V- Builder(TM), a groundbreaking tool which enables developers to easily design voice sites by creating voice interfaces to Web sites and other applications. Together with Nuance Voyager(TM), a revolutionary voice browser and Nuance's market-leading speech recognition and voice authentication software. V-Builder will enable any company to provide information and sell products in an automated fashion via an ordinary telephone.

V-Builder is unlike any other development tool because it enables developers to create voice sites in a few hours without speech recognition or telephony skills and without modifying Web pages. These voice sites can be accessed through Voyager and can be interconnected on an emerging "Voice Web" through spoken hyperlinks or "voice links." The tool also can be used to quickly develop voice interfaces for any other application for access over the telephone.

"V-Builder provides developers with the perfect tool for rapidly building voice sites," said Ronald Croen, president and CEO of Nuance. "Any company can now quickly extend the reach of their business in a very-cost effective manner to the more than 1.3 billion telephone users worldwide."

"Travelocity.com is focused on providing our more than eight million members with new and innovative ways to make their travel plans," said Terrell B. Jones, president of Travelocity.com. "V- Builder enables us to quickly provide telephone access to our travel- related products and services currently available on the Web and enables us to reach a broader audience of Voyager users."

V-Builder -- Quickly and Easily Build Voice Sites

V-Builder uses a combination of Nuance's SpeechObjects, a set of standards-based reusable application components, and the VoxML (or the forthcoming VoiceXML) programming languages, to generate voice interfaces that are accessible directly via the telephone or through the Voyager voice browser. The tool is designed so that savvy speech developers can easily integrate their applications with the Web, and Web developers can easily extend their content for access over the telephone.

The tool enables a voice interface to be mapped to fields on a Web page through the use of SpeechObjects and simple "drag and drop" commands. With V-Builder, no custom database integration is required since all the data is fed to the voice interface, and returned from the interface through the same Web (HTTP) server that services the Web site. If a Web page changes, the voice interface will still run. This enables companies to leverage their Web investment and apply it to phone users.

Companies that want to build voice sites that rely on data from non-Web sources can also reap tremendous benefits by leveraging V- Builder. The tool enables the development of voice interfaces to any packaged or custom application.

V-Builder also enables easy customization of SpeechObjects and provides a link for creating new SpeechObjects. Rapid testing of voice interfaces during the development cycle can be performed directly on the PC using a microphone. V-Builder is designed to integrate with leading Web authoring, interactive voice response (IVR) and speech recognition tools.

"As the pioneer in online trading, Charles Schwab is constantly looking for ways to extend our market leadership," said Cecily Baptist, vice president of voice technologies for Charles Schwab & Company. "Providing a voice interface to our Web-based brokerage system will help us maintain our competitive edge. We expect V- Builder to enable us to do this with minimal effort."

Pricing and Availability

V-Builder will begin beta testing in January and will be available to members of the Nuance Developer Network at no charge.

About Nuance

Nuance is the leader in natural voice interface software that makes access to information, transactions and services over the telephone convenient and secure. Every day, millions of people place calls, make travel reservations, trade stocks and interact with other telecommunications, enterprise and Web- based systems running Nuance's speech recognition, language understanding and voice authentication software. American Airlines, Bell Atlantic, Charles Schwab, The Home Shopping Network, Lloyds TSB, Sears, United Parcel Service and many other leading companies use Nuance software to provide better customer service while dramatically lowering costs. Nuance is a founding member of the V-Commerce Alliance and has led the industry in the creation of open standards for speech application development. Nuance is headquartered in Menlo Park, Calif. with global sales offices and partners supporting multilingual solutions around the world. To experience Nuance's state-of-the-art speech technology, call 888-NUANCE-8 or visit the company's Web site www.nuance.com.

NOTE: Nuance, V-Builder, V-Commerce and Voyager are trademarks of Nuance Communications. All other trademarks are properties of their respective owners. SOURCE Nuance Communications

[Reference]

Message No: Industry: COMPUTER/ELECTRONICS; TELECOMMUNICATIONS; INTERNET MULTIMEDIA ONLINE;

Indexing (document details)

Dateline: New York

Publication title: PR Newswire. New York: Oct 5, 1999. pg. 1

Source type: Wire Feed

ProQuest document ID: 45333403

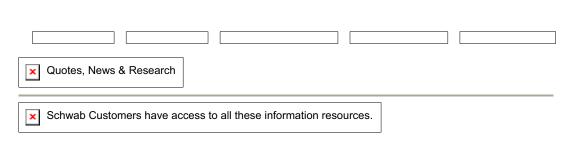
Text Word Count 777

Document URL: http://proquest.umi.com.ezproxy.apollolibrary.com/pqdweb?did=45333403&sid=-1&Fmt=3&cli

entId=13118&RQT=309&VName=PQD

Copyright © 2008 ProQuest LLC. All rights reserved.





FREE to visitors

Delayed Quotes and Charts

Access 20-minute delayed quotes. Plus intraday, daily and weekly price charts.

Market BuzzTM

News, research and opinion from a variety of independent, third-party sources.

Mutual Fund OneSource® Online

View and compare any of the 1,400 funds available through Schwab.

Fixed Income

Yield reports.

SchwabFunds®

Review performance data, and research which of their time-tested investment strategies may be best for you.

FREE to Schwab customers by logging on to the <u>Customer Center</u>.

Company News & Research

Get the latest news and research reports on the companies of your choice from independent sources.

Market News

Access in-depth, timely coverage of the markets and economy from independent sources.

More Research (fee based)

Research on RequestTM

Reports are available on over 14,000 publicly traded U.S. stocks and mutual funds.

©1997 Charles Schwab & Co., Inc. Member SIPC/New York Stock Exchange, Inc. (1297)